Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

HERRAMIENTA PARA MEJORAR LA PRESENTACIÓN DE CONTENIDOS MATEMÁTICOS EN EL AULA

Gabriela Sandoval Pantoja
Tutor: Dr. Luis Fernando Lago Fernández

Enero 2016

HERRAMIENTA PARA MEJORAR LA PRESENTACIÓN DE CONTENIDOS MATEMÁTICOS EN EL AULA

Autor: Gabriela Sandoval Pantoja
Tutor: Dr. Luis Fernando Lago Fernández

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Enero 2016

Agradecimientos

A ti, por interesarte.

Abstract

In subjects with mathematical workload, teachers often prefer using a blackboard to build algebraic reasoning because it allows them to actively explain the method of resolution. In this way, the attention of the students is captured and held during the development of the problem.

However, the use of the board has two main problems. Teachers have to copy every single step during the procedure of solving the equation, which may become a tedious assignment when working with large developments. Students focus on writing on their notebook what the teacher writes on the board, paying less attention to the explanation.

The aim of this project is the creation of ChalkPy, a tool that promotes the dynamic development of first-degree polynomial equations with a single solution through its projection to the classroom from a computer, combining the positive aspects of electronic gadgets and the use of the blackboard.

ChalkPy solves the key problems: it allows the user to interact dynamically with the equation and it keeps track of the steps performed on the equation with the option of exporting them into a document, so students do not have to make any notes. These are the features that the user can execute on the equation: moving the terms (1), calculating (2), getting common factor (3) and deleting it (4). Each one of them represents a new feature developed with SymPy, a Python's library.

The project has followed an iterative incremental life cycle with eight iterations. Django has been used as a framework for developing the web application, which follows the architectural pattern Model-Template-View (MTV). *Model* contains the information of ChalkPy in a SQLite database. The *templates* define HTML graphical interfaces dynamically generated, and it is important to highlight the use of LaTeX and MathML for rendering equations. The *views* handle navigation in the application and serve as agents between the model and the templates.

ChalkPy's success has been consolidated during a test to students in the Master's Degree in Teacher Training in Secondary Education and High School, with Math as subject, where it was introduced as a tool for teaching innovation and got good reviews.

Key words — web application, educational technology, algebra, teaching innovation project

Resumen

En asignaturas con carga de contenido matemático, los docentes suelen utilizar la pizarra para construir desarrollos algebraicos porque les permite explicar de forma activa el método de resolución a través de la interacción con ella. De esta manera se capta y mantiene la atención del estudiante y se le hace partícipe en el desarrollo del problema. Sin embargo, el uso de la pizarra tiene dos problemas principales: se vuelve una labor tediosa para los docentes el escribir los pasos de resolución de la ecuación y los estudiantes se centran en copiar de la pizarra, disminuyendo su atención.

El presente Trabajo de Fin de Grado ha permitido la implementación de ChalkPy, una herramienta que promueve el desarrollo dinámico de ecuaciones polinómicas de primer grado con solución única a través de su proyección en el aula desde un ordenador, aunando los aspectos positivos de las presentaciones electrónicas y el uso de la pizarra.

La funcionalidad que aporta ChalkPy soluciona los problemas que motivan este proyecto: permite al usuario interactuar de forma dinámica con la ecuación y registra los pasos realizados con la finalidad de exportarlos a un documento para que los estudiantes no copien apuntes.Las acciones con las que el usuario puede interactuar sobre la ecuación son: mover sus términos (1), operar (2), sacar factor común (3) y eliminarlo (4). Cada una representa una funcionalidad implementada a partir de la biblioteca SymPy de Python.

El desarrollo del proyecto ha seguido un ciclo de vida iterativo incremental con ocho iteraciones. Se ha utilizado Django como framework para desarrollar la aplicación web, que sigue un patrón de arquitectura Model-View-Template (MTV). El componente *modelo* contiene la información en una base de datos en SQLite. Los *templates* definen la interfaz gráfica generando HTMLs dinámicamente, donde destaca el uso de LaTeX y MathML para la representación de las ecuaciones. Las *vistas* manejan la navegación en la aplicación y sirven como intermediarias entre el modelo y los templates.

El éxito del proyecto se ha consolidado durante la prueba realizada a los alumnos del Máster Universitario en Formación de Profesorado de Educación Secundaria Obligatoria y Bachillerato de la rama de Matemáticas, en la cual se introdujo ChalkPy como herramienta de innovación docente y obtuvo una buena crítica.

Palabras clave — aplicación web, tecnología educativa, álgebra, proyecto de innovación docente

Glosario

Ecuación Igualdad matemática entre dos expresiones algebraicas, denominadas miembros, en las que aparecen elementos conocidos –datos– y desconocidos –incógnitas–, relacionados mediante operaciones matemáticas. XV, 1–3, 6, 7, 9–13, 15, 16, 18–21, 24–31, 34–36, 38, 39, 47, 48, 50, 51, 55–59, 65–67, 69

Equivalente Que tiene la misma solución.. 25–27, 55, 57

Factor Divisor o divisores de un número. 10, 11, 27, 57–59

Grado Exponente al que se encuentran elevadas las incógnitas.. 2, 9, 39, 55, 56

Incógnita Valor tal que, al ser sustituído, se verifica la igualdad de la ecuación.. 16, 18, 25, 29, 56

Miembro Uno de los lados de la igualdad.. 10, 11, 21, 27, 35, 50, 55, 56, 58

Solución Solución de la ecuación. 2, 9, 16, 19, 34–36, 56

Término Parte de una expresión formada por una variable y/o un coeficiente.. 7, 10, 11, 15, 20, 21, 27, 29, 48, 50, 55–59

Acrónimos

HTTP Hypertext Transfer Protocol. 31

ORM Object-Relational Mapping. 31

URL Uniform Resource Locator. 31

Índice general

Índice de tablas

Índice de figuras

1 Introducción

Este Trabajo de Fin de Grado, realizado como parte de un Proyecto de Innovación Docente de la Universidad Autónoma de Madrid, tiene como propósito el desarrollo de una herramienta para mejorar la presentación de contenidos matemáticos en el aula.

A continuación se explica la motivación y los objetivos principales del mismo, seguidos de una descripción de la estructura del presente documento.

1.1. Motivación

Es una práctica habitual que la dinámica de aprendizaje en clases con alto contenido matemático esté basada en una explicación teórica abstracta, seguida de varios ejemplos de su aplicación y la resolución de diversos ejercicios para afianzar los conocimientos. Durante este proceso, idealmente, el docente hace uso de la pizarra para explicar el método de resolución del problema mientras los estudiantes prestan atención a sus explicaciones y toman apuntes.

La pizarra es un elemento clave porque aporta dinamismo a la clase. El docente la utiliza como apoyo durante la explicación de contenidos matemáticos como las ecuaciones porque le permite explicar de forma activa el método de resolución a través de la interacción con ella. De esta manera se capta y mantiene la atención del estudiante y se le hace partícipe en el desarrollo del problema.

Sin embargo, el uso de la pizarra también tiene sus desventajas:

- La sensibilidad a posibles errores por parte del docente durante el desarrollo del problema. Por ejemplo, arrastrar un error en un signo cometido al principio de la

explicación a lo largo del proceso de resolución.

- La estructura del contenido puede volverse caótica en explicaciones extensas, lo cual lleva al estudiante a un estado de confusión.
- A veces se tiende a borrar y sobrescribir parte de las ecuaciones para evitar copiar todo de nuevo, pues es un trabajo tedioso para el docente. Sin embargo, esto supone un esfuerzo extra por parte del estudiante, que debe estar pendiente de los cambios realizados en cada iteración.
- La caligrafía del docente es un factor clave para el éxito de su explicación y el de la toma de apuntes del estudiante.
- En ocasiones se pierde demasiado tiempo en escribir pasos intermedios de baja importancia, lo cual rompe el flujo natural de las explicaciones del docente.
- Una vez borrada la pizarra, la información contenida en ella se pierde, por eso los estudiantes se centran más en copiar lo que está escribiendo el docente en ella que en comprender y afianzar la explicación que está desarrollando.

La motivación de este Trabajo de Fin de Grado es dar solución a estos inconvenientes manteniendo las ventajas del uso de la pizarra a través de una herramienta interactiva que permita el manejo de ecuaciones de forma dinámica: ChalkPy.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es el diseño y desarrollo de ChalkPy, una herramienta que permite el desarrollo dinámico de ecuaciones polinómicas de primer grado a través de su proyección en el aula desde un ordenador, aunando los aspectos positivos de las presentaciones electrónicas y el uso de la pizarra.

Este objetivo se considerará alcanzado si se cumplen las siguientes condiciones:

- El ámbito de las expresiones algebraicas con las que trabajar se centra en las ecuaciones polinómicas de primer grado con solución única (ver alcance en sección 3.1).
- Las ecuaciones se manipulan de forma interactiva mediante únicamente el uso del ratón.
- Se almacena un registro de las manipulaciones realizadas sobre la ecuación.
- Permite generar un documento que refleje las manipulaciones realizadas sobre la ecuación.
- Permite la resolución de la ecuación de forma automática, con la indicación de los pasos realizados para obtener la solución.

- El usuario puede definir las ecuaciones a utilizar, con la posibilidad de utilizar variables definidas por él.

1.3. Estructura del documento

El presente documento tiene la siguiente estructura:

- En el capítulo 2 se define el estado del arte con el objetivo de exponer al lector en situación respecto al uso de nuevas metodologías en el aula, especialmente aquellas relacionadas con el ámbito de ChalkPy: el álgebra.
- El capítulo 3 especifica el alcance de la aplicación, define la metodología aplicada en el desarrollo de este Trabajo de Fin de Grado y las herramientas y tecnologías que se han utilizado.
- El capítulo 4 describe el análisis a través de la enumeración de los requisitos funcionales y no funcionales del proyecto.
- En el capítulo 5 se explica el diseño que sigue la aplicación, definiendo su patrón de arquitectura y especificando sus componentes.
- El capítulo 6 entra en detalle sobre cómo se ha implementado la aplicación siguiendo una estructura similar a la mencionada en el capítulo anterior.
- En el capítulo 7 se definen las pruebas que se han realizado sobre la aplicación durante y tras su desarrollo.
- El capítulo 8 contiene la conclusión del proyecto y las posibles líneas de trabajo futuro a raíz de todo su desarrollo.

2

Estado del arte

En este capítulo se pone al lector en la situación a la que se han enfrentado los docentes para exponer sus conocimientos a lo largo de los años para introducirle después a las nuevas tecnologías que conviven junto a ellos en las aulas actualmente. Estas tecnologías han supuesto la introducción de nuevas herramientas, de las cuales se mencionan especialmente aquellas relacionadas con las matemáticas y, reduciendo más el ámbito, aquellas relevantes para este Trabajo de Fin de Grado relacionadas con el álgebra lineal.

2.1. Introducción

El primer contacto encontrado del uso de tiza y pizarra como instrumentos de enseñanza se remonta a una escuela de la India en el siglo XI, mencionado por Abũ 'r-Raihãn Muhammad ibn Ahmad al-B̃irũñi en su obra *Un estudio crítico de lo que la India dice, bien sea racionalmente aceptado como refutado*, también conocido como *India* [1].

> *"Utilizan tablas negras para los niños en las escuelas, y escriben en ellas a lo largo del lado largo, no del lado ancho, escribiendo con un material blanco de izquierda a derecha."* (Traducción no oficial)

Con esta cita se puede afirmar que al menos diez siglos llevan pizarra y tiza formando parte del material que los docentes utilizan para hacer llegar a los estudiantes sus conocimientos.

Sin embargo, durante todo ese tiempo, ambas han ido evolucionando y adaptándose a las nuevas necesidades de sus usuarios. Los cambios más actuales han supuesto desde

la sustitución de la pizarra como material por el plástico y la tiza por rotuladores, hasta llegar a la digitalización de la misma.

Actualmente se pueden encontrar en las aulas de colegios, institutos, universidades y demás centros de enseñanza elementos como ordenadores, proyectores y pantallas de proyección, o también pizarras digitales, que permiten la interacción sobre la misma con un bolígrafo especial —o incluso los propios dedos— para manejar el contenido.

2.2. Herramientas matemáticas

La evolución de las herramientas para la enseñanza conlleva también la evolución de las metodologías a aplicar. Con la introducción de nuevas tecnologías en las aulas, los docentes han adaptado la presentación del contenido de su materia de forma que tanto la transmisión de sus conocimientos como su comprensión por parte de los alumnos sea más dinámica y efectiva.

En el ámbito de las matemáticas se pueden encontrar diversas herramientas para que docentes de esta disciplina y estudiantes complementen su enseñanza y aprendizaje, respectivamente. Una selección de las más interesantes se muestra en el Anexo B.

Se puede observar que estas aplicaciones, en gran medida, intentan abarcar todo el área de las matemáticas que se enseñan en el aula y están enfocadas principalmente a cubrir las necesidades del alumno, tanto durante las clases como fuera del aula. También se mencionan algunas más enfocadas al área de interés de este Trabajo de Fin de Grado, el álgebra, pero no comparten su objetivo.

Las herramientas más similares a lo que se espera de este Trabajo de Fin de Grado se describen a continuación.

Mathway

Mathway [2] es una herramienta web para la resolución de ejercicios de cálculo, geometría, estadística y álgebra, entre otras muchas áreas.

Su sección de álgebra cubre uno de los objetivos de este Trabajo de Fin de Grado, pues permite al usuario introducir una expresión algebraica que el programa resuelve y te explica en líneas generales cómo se debe hacer.

Sin embargo, para ver la resolución paso a paso es necesario registrarse como usuario. Tampoco cumple el objetivo principal, que es la manipulación de las ecuaciones por el usuario.

También está disponible para móviles en Play Store (Android), App Store (iOS) y Amazon Appstore.

Mathination

Mathination [3] es una aplicación móvil desarrollada para iOS que tiene la característica clave que quiere conseguirse con el presente Trabajo de Fin de Grado, pues permite la manipulación de ecuaciones dinámicamente.

En esta manipulación destaca el movimiento de términos por la ecuación deslizándolos con el dedo y su cálculo pellizcándolos.

Cuenta con una base de datos en la que se encuentran problemas ya predefinidos y además ofrece ayuda al usuario durante la resolución del ejercicio, en caso de que sea necesaria.

Esta aplicación cubre tanto las necesidades del profesor para explicar el desarrollo de una expresión matemática en clase como las del alumno, que puede utilizarla fuera del aula para practicar.

Su mayor inconveniente es que sólo está disponible para una plataforma móvil (iOS) y tiene un coste asociado.

Algebra Touch

Algebra Touch [4], similar a Mathination, es una aplicación móvil que cubre un sistema operativo más, pues está disponible tanto en App Store (iOS) como en Windows Store.

Cabe destacar su manipulación para operar términos, pues se enfoca de forma distinta a la de la aplicación anterior. En este caso es necesario tocar el símbolo de la operación para que esta se efectúe.

Tiene también un módulo para explicar la resolución del ejercicio durante su desarrollo, una base de datos con problemas ya predefinidos y da la posibilidad de guardar ejercicios personalizados.

Al igual que en el caso anterior, su mayor inconveniente es su coste asociado y que no está disponible para la gran mayoría de usuarios de dispositivos móviles, puesto que no cubre el sistema operativo Android.

2.3. Conclusiones

Tras la inmersión en las diversas herramientas matemáticas que se encuentran actualmente disponible se observa que abarcan una gran variedad de ámbitos, objetivos y plataformas. Entre ellas destacan tanto juegos como herramientas de apoyo, cuya finalidad principal es proporcionar al alumno un medio con el que aprender y mejorar sus habilidades en diferentes áreas.

A pesar de esta variedad, no abundan las herramientas desarrolladas explícitamente

para que el docente las utilice como parte de su metodología. Algunas, como las mencionadas *Algebra Touch* y *Mathination*, sí pueden ser utilizadas como apoyo en el aula. Sin embargo, ambas están restringidas por su disponibilidad y su coste.

ChalkPy pretende reunir los aspectos dinámicos de estas aplicaciones y corregir sus limitaciones como aplicación web de código abierto que permita una fácil inclusión de nuevos contenidos y funcionalidades conforme avance su desarrollo, así como promover el uso de tecnologías en el aula como apoyo para los docentes.

3

Definición del proyecto

En este capítulo se explica cuál es el alcance de este ChalkPy, la metodología aplicada en su desarrollo y las herramientas y tecnologías que se han utilizado.

3.1. Alcance

ChalkPy permite la resolución de ecuaciones polinómicas de primer grado con solución única compuestas por las siguientes operaciones: sumas, restas, multiplicaciones y divisiones. Estas operaciones se implementarán de forma escalable para la inclusión de nuevos contenidos y funcionalidades en la aplicación.

El público objetivo de ChalkPy son los docentes de un nivel correspondiente a 1º de la ESO según el sistema educativo español. En dicho nivel los estudiantes tienen contacto por primera vez con este área de las matemáticas y se pretende que ChalkPy promueva su interés por el álgebra a través de su uso como parte de una metodología dinámica en el aula.

3.2. Metodología

Partiendo de un conocimiento poco afianzado de los recursos a utilizar para el desarrollo de la herramienta, tras la declaración de sus objetivos y la restricción de su alcance se observó que la metodología ágil más apropiada a utilizar en el desarrollo es aquella que sigue un ciclo de vida iterativo e incremental.

Este tipo de metodología aúna las ventajas de los ciclos de vida en cascada con las del modelo evolutivo, identificando las funcionalidades principales de la herramienta y desarrollándolos según sus prioridades [5]. Por ello, el ciclo de vida iterativo e incremental utilizado permite obtener resultados tras cada iteración con los que experimentar e ir ampliando sus funcionalidades poco a poco.

Las etapas llevadas a cabo durante cada iteración han sido las siguientes:

1. Análisis de los requisitos.
2. Prototipado que refleja la funcionalidad a alcanzar en la iteración.
3. Implementación.
4. Pruebas sobre la funcionalidad implementada.
5. Revisión de la iteración y planteamiento de la siguiente.

Estas etapas reflejan el ciclo de vida en cascada y permiten obtener un prototipo que irá evolucionando a medida que se incremente la funcionalidad con cada iteración.

A continuación se indican las iteraciones llevadas a cabo. Excepto el análisis inicial de requisitos, todas las iteraciones siguen las etapas mencionadas anteriormente, por lo que sus entradas y salidas coinciden con la aplicación en sí. Por ello, sólo se van a mencionar las tareas realizadas en cada una de ellas.

- **Análisis inicial de requisitos**.
  - Tareas:
    - Definición de los objetivos de la aplicación.
    - Investigación de las operaciones básicas en la resolución de ecuaciones.
    - Investigación de la biblioteca SymPy y práctica en el entorno que proporciona su consola online [6].
  - Salida:
    - Ejemplos de varias ecuaciones resueltas paso a paso.
    - Lista de funcionalidades a tener en cuenta al desarrollar ChalkPy (se puede consultar en el Anexo C).
- **Iteración 1: Funcionalidad mover**. Implementar la funcionalidad mover, que permite cambiar de posición los términos de una ecuación, tanto en un mismo miembro como de uno a otro.
- **Iteración 2: Funcionalidad simplificar**. Implementar la funcionalidad simplificar, que permite operar los términos de una ecuación, cubriendo sumas, restas, multiplicaciones y divisiones.
- **Iteración 3: Funcionalidad factor común**. Implementar la funcionalidad factor común, que permite obtener el factor numérico común entre varios términos.

- **Iteración 4: Funcionalidad eliminar factor común**. Implementar la funcionalidad eliminar factor común, que permite eliminar el término que se repite en ambos miembros de la ecuación.

- **Iteración 5: Implementación de pruebas**. Implementar pruebas específicas para las funcionalidades desarrolladas en las iteraciones anteriores para la corrección en profundidad de su funcionalidad. Incluye la implementación de resolución automática de ecuaciones.

- **Iteración 6: LaTeX**. Investigar e implementar la funcionalidad de generar un documento LaTeX que contenga los pasos realizados durante la resolución de una ecuación.

- **Iteración 7: Aplicación web**. Implementar la interfaz de la aplicación a través de Django: funcionalidades mover, simplificar, factor común y eliminar factor común.

- **Iteración 8: Ampliación de la funcionalidad de la aplicación web**. Ampliar la funcionalidad de la aplicación: crear y eliminar ecuaciones, resolver ecuación, navegar por los pasos de la ecuación, generar documento PDF.

3.3. Tecnologías y herramientas utilizadas

En esta sección se definen las tecnologías y herramientas utilizadas durante el desarrollo de este Trabajo de Fin de Grado y los ámbitos en las que se han visto implicadas.

Tecnologías

Las tecnologías utilizadas en el desarrollo de ChalkPy son las siguientes:

- Python [7] como lenguaje de programación. Su elección se ha visto influenciada por el uso de la biblioteca de matemática simbólica SymPy [8].

- Django [9] como framework de desarrollo web. En los capítulos5 y 6 se entrará en detalle en sus características.

- SQLite [10] como base de datos, pues forma parte de Python. En la sección 5 se habla más sobre ella.

- LaTeX [11] y MathML [12] para la representación de las ecuaciones. Ver secciones 6.3.2 y 6.3.3, respectivamente.

- HTML [13] para la elaboración de las páginas web. Ver sección 6.3.1.

- JavaScript [14] y JQuery [15] como parte de la interfaz gráfica. Ver sección 6.3.4.

Herramientas

Entorno

Durante las seis primeras iteraciones se ha utilizado el editor de texto Sublime Text [16] para su implementación, debido al pequeño tamaño del proyecto. La gran ventaja que aporta es que, gracias a su gran variedad de prestaciones, la escritura y especialmente la modificación del código durante su desarrollo se vuelven tareas mucho más ligeras.

Al comenzar a implementar la aplicación web, se eligió PyCharm [17] como entorno de trabajo. PyCharm es el IDE de JetBrains con el que programar en Python y permite la creación de aplicaciones web con Django. Sus prestaciones son extraordinarias.

Bibliotecas externas

Para el desarrollo de este proyecto se han utilizado las siguientes bibliotecas externas de código abierto:

- *SymPy* [8] es una biblioteca de Python para trabajar con matemática simbólica. Esto significa que sus objetos matemáticos se representan exactamente, no de forma aproximada, lo que supone que las expresiones con variables no evaluadas se expresen de forma simbólica. Esta biblioteca fue el motivo principal por el que se decidió construir la aplicación en Python, pues contiene funcionalidades muy útiles en el manejo de ecuaciones algebraicas. En la sección 6.2.2 se entrará en detalle sobre su uso en la aplicación.
- *PyLaTeX* [18] es una biblioteca de Python que permite crear y compilar archivos en LaTeX. A través de ella se consigue la generación del documento PDF que contiene los pasos realizados para resolver una ecuación.

Control de versiones

El control de versiones sirve para gestionar los cambios realizados durante la creación de un proyecto. Se ha utilizado la plataforma GitHub [19] —basada en el software Git— como plataforma en la que almacenar los archivos, pues ofrece un plan privado gratuito para estudiantes. Además, está integrada en PyCharm, lo cual facilita la interacción con los archivos de la plataforma desde el propio entorno.

Pruebas

Durante el periodo de validación (ver sección 7.2.1) se decidió realizar pruebas sobre una pizarra digital para conocer la usabilidad de la aplicación sobre ella, pues comienzan a forman parte de las aulas. Se utilizó el modelo de pizarra digital SmartBoard M680, de

SmartTech, disponible en el aula multimedia del Edificio de Posgrado de la Universidad Autónoma de Madrid.

Gráficos

Para la creación de las imágenes que contiene el presente documento se han utilizado diversas aplicaciones:

- *Dia* [20], para el diagrama E-R 5.2 de la base de datos.
- *Balsamiq* [21], para las maquetas de la aplicación. Ver apartado 5.3.1.
- *Cacoo* [22], para dibujar los árboles de ecuaciones.
- *Gimp* [23], como herramienta de edición general.

4
Análisis

En este capítulo se presenta el análisis de ChalkPy reflejado en sus requisitos funcionales y no funcionales.

Estos requisitos definen las características finales de la aplicación, pues debido a la metodología utilizada a lo largo del proceso de desarrollo ha habido una constante modificación y ampliación de los mismos.

4.1. Requisitos funcionales

Los requisitos funcionales son aquellos relacionados con la ejecución de la herramienta. Se resumen en:

RF. 1 La ecuación podrá resolverse a partir de la interacción con los términos a través de las siguientes funciones: mover, simplificar, obtener factor común y eliminar factor común. Consultar anexo C para el análisis de las funcionalidades.

RF. 2 El orden de los términos no se modificará durante la interacción con la ecuación.

RF. 3 No se podrán realizar acciones incorrectas sobre la ecuación.

RF. 4 La interfaz mostrará una lista de los pasos realizados sobre la ecuación.

RF. 5 Cada paso realizado tendrá asociado un comentario que indique la funcionalidad aplicada respecto al paso anterior.

RF. 6 El usuario podrá modificar los comentarios asociados a los pasos realizados.

RF. 7 La solución de la ecuación se podrá obtener de forma automática.

RF. 8 Los pasos creados durante la resolución automática de una ecuación se generarán con un comentario descriptivo sobre la funcionalidad aplicada en cada uno de ellos.

RF. 9 Será posible crear un documento PDF que refleje los pasos realizados durante la resolución de la ecuación.

RF. 10 Se podrá navegar por lo pasos de la ecuación, consultando los pasos previos y pudiendo volver al último realizado. En caso de realizar una nueva acción en un paso anterior al último, los pasos posteriores al actual serán eliminados.

RF. 11 La interacción con la ecuación podrá ser reiniciada, volviendo a la ecuación inicial.

RF. 12 El usuario podrá añadir y eliminar ecuaciones.

RF. 13 El usuario podrá definir variables propias con las que trabajar como incógnitas.

RF. 14 Se indicará al usuario qué ecuaciones de las que ha definido ha resuelto.

4.2. Requisitos no funcionales

Los requisitos no funcionales están relacionados con la interfaz, la usabilidad y la documentación, entre otras cosas. Son:

RNF. 1 La interacción con los elementos de la interfaz se podrá realizar únicamente a través de puntero, a excepción de la entrada de teclado para definir las ecuaciones y la modificación de los comentarios a las acciones.

RNF. 2 El programa se podrá utilizar en el navegador Firefox a partir de la versión 43.

RNF. 3 Al menos una interacción con la ecuación durante su resolución deberá ser realizada mediante el método Drag&Drop.

RNF. 4 Se utilizará LaTeX para mostrar las ecuaciones como texto.

RNF. 5 El tiempo de respuesta en la interacción con la ecuación será menor a 1s. En el caso de la resolución automática, se permite una demora de hasta 5s.

RNF. 6 La interfaz gráfica de la aplicación tendrá sus textos en inglés.

RNF. 7 La herramienta contará con un Manual de Usuario (ver Anexo A).

5 Diseño

En este capítulo se explica el diseño de ChalkPy con el objetivo de satisfacer los requisitos especificados en el capítulo 4. Para ello se define el patrón de arquitectura que se ha seguido y se entra en detalle en los aspectos de diseño de sus componentes.

5.1. Patrón de arquitectura

Debido al uso de Python como lenguaje de la aplicación, el framework con el que se ha se ha decidido trabajar es Django. Django utiliza una adaptación del patrón de arquitectura Modelo-Vista-Controlador (MVC) [24], conocida como Model-Template-View (MTV) [25].

En ella, la vista describe la información que se presenta al usuario. Por su parte, el template (*plantilla*) es quien describe cómo los datos son presentados. En la figura 5.1 se puede apreciar la relación entre los componentes de cada patrón.

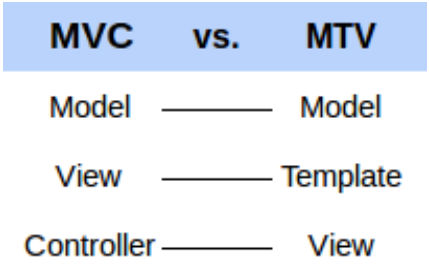

Figura 5.1: Relación entre patrones MVC y MTV

5.2. Model

El modelo está definido por una base de datos, cuyas entidades y atributos se definen en el archivo *models.py*.

La base de datos está compuesta por las ecuaciones con las que se interactúa, los pasos que se siguen hasta resolverla y las incógnitas propias que puede utilizar el usuario, relacionadas como se muestra en la figura 5.2.

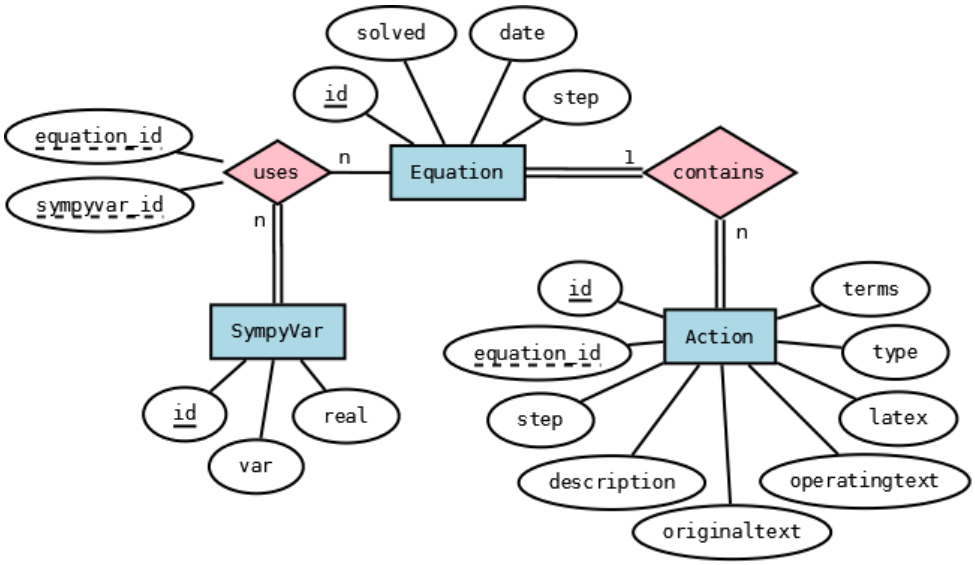

Figura 5.2: Modelo Entidad-Relación de la base de datos

Como se ha mencionado en el capítulo 4, el usuario tiene la posibilidad de crear ecuaciones con incógnitas especiales a las que vamos a denominar *incógnitas propias*. Se consideran especiales porque van más allá de las incógnitas clásicas –*x,y,z*– y pueden definir expresiones complejas como "P(A|B)". La necesidad de diferenciarlas se debe a los problemas que surgen entre su uso como incógnitas y la funcionalidad que ellas mismas representan cuando se trabaja con ellas en SymPy y LaTeX.

Para trabajar con ellas se utiliza la entidad `SympyVar`, que relaciona la expresión que el usuario quiere utilizar con una variable auxiliar que será utilizada internamente como su sustituta cuando se trabaje con la ecuación.

- `SympyVar`. Relaciona una variable definida por el usuario con su variable operativa. Sus atributos son:
  - `id`. Identificador de `SympyVar`.
  - `var`. Variable auxiliar con la que trabaja una ecuación.
  - `real`. Variable definida por el usuario.

La entidad `Equation`, que define la ecuación del usuario, se relaciona de forma n-m

con SympyVar a través de la tabla `Equation_SympyVar`, que contiene la clave primaria de ambas.

- `Equation`. Entidad principal de la aplicación. Sus atributos son:
  - `id`. Identificador de la ecuación.
  - `step`. Número de acciones que se han realizado en la ecuación.
  - `solved`. Indica si la ecuación ha sido resuelta.
  - `date`. Fecha en la que se creó la ecuación.
- `Equation_SympyVar`. Relaciona `Equation` con `SympyVar`.
  - `equation_id`. Identificador de la ecuación.
  - `sympyvar_id`. Identificador de `SympyVar`.

Sobre las ecuaciones se realizan acciones que van alterándolas hasta llegar a la ecuación equivalente que indica su solución. Cada objeto de la entidad `Action` define la funcionalidad que se ha aplicado sobre una ecuación, siendo su relación con ella 1-n.

- `Action`. Funcionalidad aplicada a una ecuación para crear una equivalente. Sus atributos son:
  - `id`. Identificador de la acción.
  - `type`. Tipo de acción a realizar. Puede ser INIT, MOVE, SIMPLIFY, COMMONFACTOR Y DELCOMMONFACTOR. Para más detalle, ver subsección 6.2.2.
  - `terms`. Sobre qué términos se realiza la acción.
  - `step`. Número de acción sobre la ecuación.
  - `description`. Descripción del paso a realizar.
  - `originaltext`. Ecuación con las variables propias del usuario.
  - `operatingtext`. Ecuación con las variables auxiliares.
  - `latex`. Ecuación en formato LaTeX.

5.3. Template

Los templates presentan la información que contiene el modelo para que el usuario interactúe con ella, es decir, definen la interfaz gráfica. El aspecto clave de esta aplicación es que debe haber una interacción dinámica con la ecuación, por lo tanto tiene especial relevancia el diseñar una página limpia, usable y descriptiva que capte la atención del estudiante durante las explicaciones del docente y que a su vez contenga todas las funcionalidades principales.

A continuación se muestran algunas maquetas de la aplicación. En el apartado 6.3 se entrará en detalle sobre cómo se implementan los templates.

5.3.1. Maquetas

En este apartado se muestran las maquetas de las dos páginas principales de la aplicación: *notebook.html* y *chalkboard.html*.

La figura 5.3 refleja la pantalla denominada *notebook.html*. En ella, el usuario visualiza todas las ecuaciones que ha definido.

- Pulsando en (1), el usuario puede añadir más ecuaciones a su cuaderno a través de un prompt.
- (2) Se muestra al usuario si ya ha resuelto esa ecuación.
- (3) Permite eliminar las ecuaciones de la lista.
- Pulsando sobre cualquiera de las ecuaciones (4) se accede a la pantalla *chalkboard.html* para trabajar con la seleccionada.

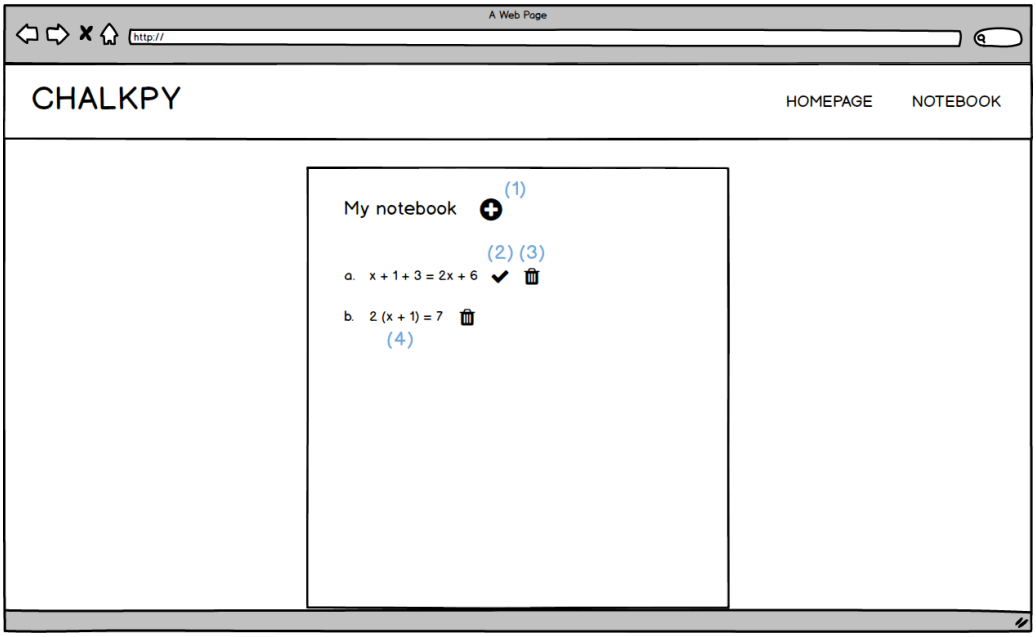

Figura 5.3: Maqueta de la página notebook

La pantalla *chalkboard.html* que se muestra en la figura 5.4 es la más importante. Esta se divide visualmente en dos áreas, A y B, que contienen diferentes elementos enfocados a la funcionalidad de cada una.

La parte A, a la izquierda, reúne los elementos a través de los cuales interactuar con la ecuación. En ella encontramos:

- La ecuación (5). Su interacción con ella se realiza pulsando y arrastrando los términos para moverlos y doble clic para seleccionar.

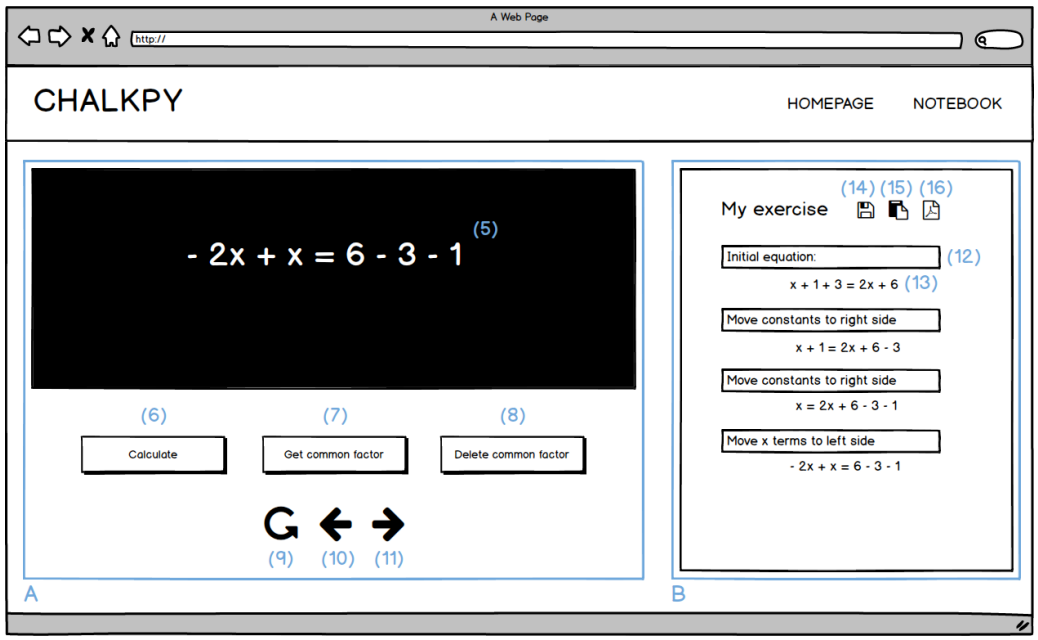

Figura 5.4: Maqueta de la página chalkboard

- El botón "Calculate" (6) permite operar los términos seleccionados.
- El botón "Common factor" (7) saca factor común de los términos seleccionados.
- El botón "Delete common factor" (8) elimina los términos seleccionados si son comunes en ambos miembros.
- La flecha (9) permite reiniciar la ecuación.
- La flecha (10) permite volver a pasos anteriores.
- La flecha (11) permite regresar a pasos posteriores que se hayan realizado.

La parte B, en la parte derecha de la pantalla, muestra un listado de las acciones que se han realizado sobre la ecuación.

- (12) muestra una descripción del paso aplicado a la ecuación que ha resultado en (13). Esta descripción puede modificarse.
- El botón (14) guarda los cambios realizados sobre las descripciones de los pasos.
- El botón (15) permite resolver la ecuación desde el paso en que se encuentre.
- El botón (16) genera un documento PDF con las acciones realizadas hasta entonces.

El motivo por el que se ha decidido centrar la mayor parte de la funcionalidad de la aplicación en esta pantalla es evitar la distracción de los estudiantes si se navega a otra parte de la aplicación para, por ejemplo, ver el registro de pasos realizados.

5.4. View

En Django se denomina *vistas* a lo que se conoce como controlador en la arquitectura MVC, y describe la información que se presenta al usuario. En el apartado 6.4, a continuación, se entrará más en detalle sobre ellas.

5.5. Conclusión

Django utiliza el patrón MTV, cuya interacción entre componentes se refleja en la figura 5.5. Este esquema introduce otros elementos que se conocerán en el capítulo 6, a continuación.

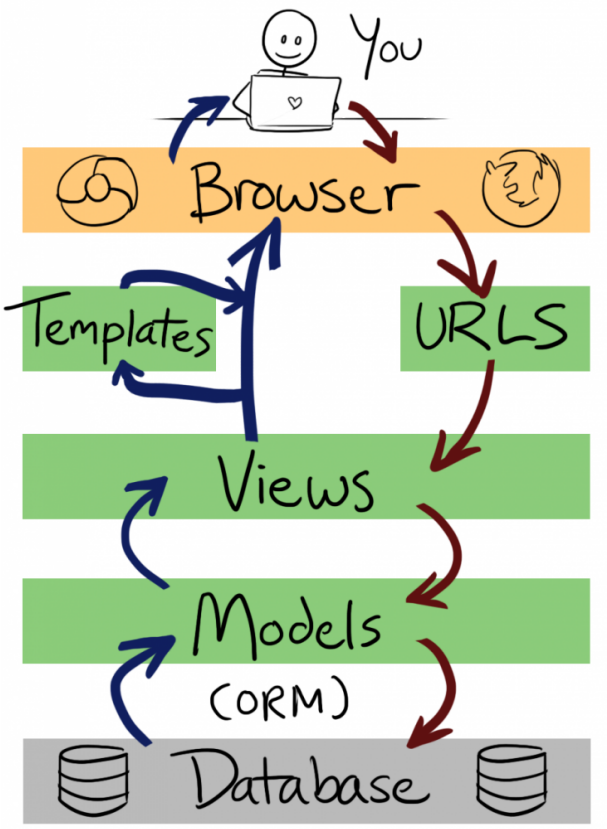

Figura 5.5: Esquema patrón Model-Template-View [26]

6 Implementación

En esta sección se explica la estructura de ChalkPy y cómo se ha implementado el patrón de arquitectura Model-Template-View descrito en el capítulo 5.

6.1. Estructura del proyecto

La disposición de archivos de ChalkPy refleja la estructura básica que toda aplicación creada con Django sigue [27], además de incluir otros recursos necesarios para el correcto funcionamiento de la aplicación.

- chalkpy/
  - `__init__.py`
  - `logic/`. Carpeta que contiene la lógica creada para la extensión de las funcionalidades de SymPy. Ver subsección 6.2.2.
  - `migrations/`. Almacena un log de las modificaciones realizadas sobre la base de datos. Ver subsección 6.2.1.
  - `models.py`. Define la base de datos. Ver subsección 6.2.1.
  - `static/`. Reúne los recursos de imágenes, estilo y los archivos js. Ver subsección 6.4.
  - `templates/`. Agrupa los archivos HTML que se utilizan en la aplicación. Ver subsección 6.3.
  - `urls.py`. Define las URL de la aplicación. Ver subsección 6.4.
  - `views.py`. Define las vistas de la aplicación. Ver subsección 6.4.

6.2. Model

En el componente modelo se engloban dos partes de la aplicación que se encargan de la gestión y manipulación de la información con la que la aplicación opera: la base de datos que contiene y la lógica que necesita para su correcto funcionamiento.

6.2.1. Base de datos

La base de datos, detallada en el apartado 5.2, está implementada en SQLite [10]. El motivo principal de esta elección es que no es necesaria una base de datos extremadamente robusta, y SQLite ya está incluida en Python [28].

Django genera y modifica automáticamente la base de datos a partir de su definición en el fichero *models.py* descrita en la sección 5.2. Cuando se realiza algún cambio en el código durante su implementación, estas modificaciones se almacenan en la carpeta *migrations.py* como archivos que describen el conjunto de cambios que se han aplicado a la base de datos —creación, modificación o eliminación de tablas—. Después se aplican dichas modificaciones a la base de datos, sincronizando los cambios realizados en el modelo.

Cada entidad de la base de datos se corresponde con una de las clases definidas en el archivo *models.py*, que se representan como subclases de la clase de Django *Model*. Por su parte, los atributos de las entidades coinciden con las variables que contiene la clase, representadas por instancias de la clase Field de Django.

En ChalkPy ha sido necesario definir únicamente tres clases: *SympyVar*, *Equation* y *Action*. No se ha mencionado *Equation_SympyVar* porque dicha tabla es generada automáticamente a través de un atributo específico en la definición de *Equation*:

```
variables = models.ManyToManyField(SympyVar)
```

La forma en que se relacionan *Equation* y *Action* es indicando en *Action* que tiene una clave foránea *Equation*:

```
equation = models.ForeignKey(Equation)
```

Para que la aplicación pueda relacionarse con la base de datos, cada clase cuenta con sus correspondientes métodos que permiten obtener y modificar los datos almacenados. Además de la funcionalidad heredada, ha sido necesario crear otros métodos para cubrir todos los aspectos de la aplicación.

En la clase *Equation* cabe destacar los siguientes:

- `newaction`. Genera un nuevo objeto *Action* a partir de los datos que recibe tras la realización de una operación sobre la ecuación en la interfaz. Para ello utiliza la lógica de *eqmove.py*, *eqsimplify.py*, *eqcommonfactor.py* y *eqdelcommonfactor.py*, que se describen en la subsección 6.2.2.
- `reload, undo, redo`. Son llamados para aplicar las acciones de recargar la

ecuación, volver al paso anterior y ver el paso siguiente, respectivamente.

- `solve`. Resuelve la ecuación creando a su paso los objetos *Action* necesarios. Para ello utiliza la lógica de *eqsolve.py*, descrita en la subsección 6.2.2.

- `savedocument`. Genera un documento PDF en LaTeX a partir de todos los objetos *Action* realizados sobre la ecuación hasta el momento. Para ello utiliza la lógica de *latexdocument.py*, descrita en la subsección 6.2.2.

Por su parte, en la clase *Action* cabe destacar el siguiente método:

- `createparts`. De forma recursiva genera una lista que contiene la información necesaria para dibujar una ecuación con MathML. Esto se explicará en profundidad en la subsección 6.3.3.

6.2.2. SymPy

Como ya se ha mencionado en la sección 3.3, SymPy [8] es una biblioteca de Python de matemática simbólica con varias funcionalidades útiles en el manejo de las ecuaciones algebraicas definidas a partir de la clase de Python `Eq` (*Equality*). Esta clase relaciona dos objetos mediante la operación de igualdad [29] y los mantiene sin evaluar si la relación entre ellos no es obvia. Esto supone para nuestra aplicación que las expresiones algebraicas que se creen como objeto `Eq` podrán manipularse.

El objeto `Eq`, como toda expresión matemática de Python, tiene varias propiedades de las cuales cabe destacar *func* y *args*. La primera indica la funcionalidad principal de una expresión, mientras que la segunda indica los argumentos a los que se aplica dicha funcionalidad. Estos elementos a su vez pueden contener más expresiones, de forma que se puede representar el objeto `Eq` como un árbol de expresiones cuyos nodos son funciones o incógnitas [30]. Se va a utilizar como ejemplo la ecuación `(2x+2)/3=5-x-1`, de la cual la figura 6.1 muestra su disposición interna en forma de árbol.

En la figura 6.1 se puede apreciar el método que utiliza SymPy para representar las expresiones matemáticas, y es que no existe clase alguna para las restas o para las divisiones, sino que en el caso de las restas se utilizan las clases Add y Mul, y para las divisiones se utilizan las clases Mul y Pow. La forma en que se representa la ecuación internamente es:

Equality(Mul(Pow(Integer(3),Integer(-1)),Add(Mul(Integer(2),Symbol('x')),Integer(2))), Add(Mul(Integer(-1),Symbol('x')),Integer(-1),Integer(5)))

Las cuatro primeras iteraciones del desarrollo de ChalkPy (ver sección 3.2) se centraron en la implementación de varias funcionalidades necesarias para la manipulación de objetos `Eq`. Todas ellas se centran en la creación de ecuaciones equivalentes a la ecuación dada tras aplicarle la funcionalidad elegida, puesto que SymPy no permite modificar el objeto `Eq`.

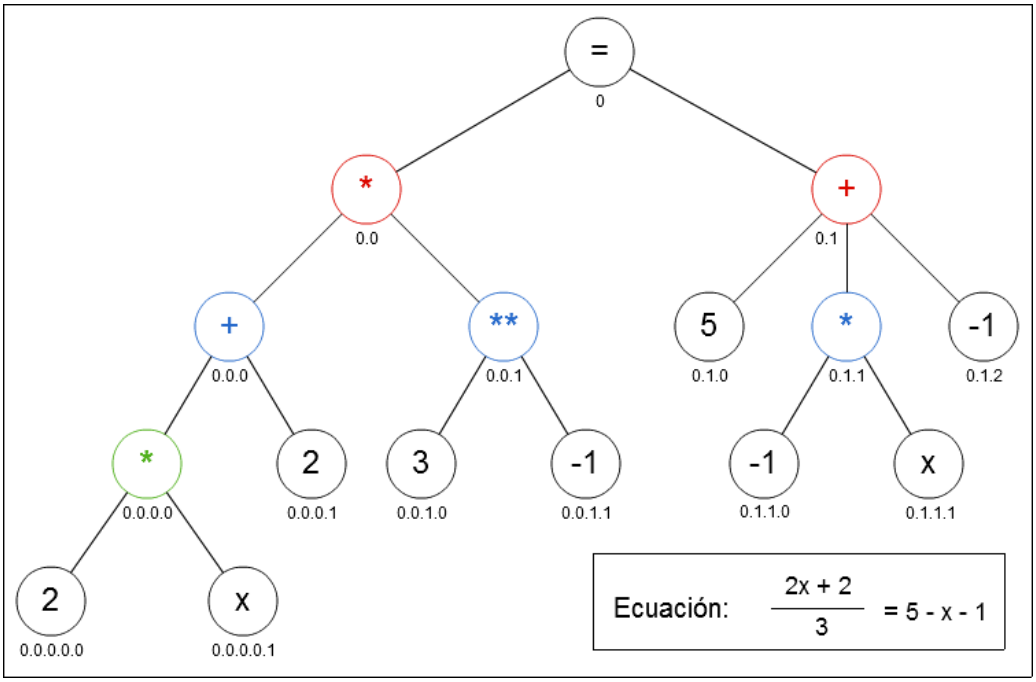

Figura 6.1: Expresión `(2x+2)/3=5-x-1` expresada como árbol

Para ello se han creado funciones nuevas a aplicar a los objetos `Eq` que devuelven la ecuación equivalente según se haya simplificado, movido, sacado factor común o eliminado factor común. En el Anexo C puede consultarse la investigación que se realizó sobre las operaciones clave a la hora de resolver ecuaciones.

Todas las acciones tienen algo en común, y es que necesitan recorrer recursivamente el árbol que forma el objeto `Eq` de forma infija. Puesto que, como se ha visto en la figura 6.1, el objeto `Eq` no es un árbol binario, es necesario adaptar el algoritmo de recursión de orden medio. Mostrando el uso de la función *args* ya mencionada, el pseudocódigo utilizado es el siguiente:

```
ordenMedio(eq){
  si len(eq.args) != 0:
    for i := 0 to len(eq.args) do:
      ordenMedio(eq.args[i])
      if i < len(eq.args)-1 do:
        visitar(eq)
  else:
        visitar(eq)
}
```

Para identificar los nodos del árbol se utilizan unos identificadores que indican la profundidad y la posición en que se encuentran. Estos identificadores son los que permiten internamente reconocer los nodos sobre los que se quiere realizar alguna de las acciones.

Por lo tanto, en el ejemplo utilizado hasta ahora, si se quiere operar `5-1`, internamente se aplicará la función *simplificar* a los identificadores `0.1.0` y `0.1.2`. En el Anexo D se muestra la resolución paso a paso de la ecuación utilizada como ejemplo indicando las acciones utilizadas y el árbol resultante de cada una.

A continuación se van a describir brevemente estas cuatro funcionalidades. Para profundizar en ellas, consultar el Anexo C.

- `move`. Siempre y cuando la acción sea posible, *move* permite mover un término de la ecuación a otra parte. Hay dos posibles formas de moverse: en el mismo miembro de la ecuación y al otro miembro de la ecuación, que se distinguen por los identificadores de origen y destino. Esta diferencia es necesaria para tener en cuenta que, al trabajar con ecuaciones, es necesario cambiar la operación de los términos que se mueven al miembro opuesto: las sumas pasan como restas y las multiplicaciones como divisiones, y viceversa. Esta funcionalidad tiene una dificultad añadida durante su implementación, y es que la ecuación que muestra SymPy no coincide muchas veces con el orden que se representa internamente como árbol, por lo que ha sido necesario crear funciones que se encarguen de mostrar los términos en el orden correcto.

- `mysimplify`. SymPy tiene una función conocida como *simplify* [31] que permite reducir una ecuación, o lo que es lo mismo, operar sus términos. En el caso de nuestra aplicación, lo que interesa no es reducir las ecuaciones al mínimo sino ir operando paso a paso los términos que se seleccionen, por lo que se ha creado mysimplify. En ella, siguiendo el estilo de la funcionalidad *move* y de forma parecida a como trabaja *simplify*, se operan únicamente aquellos términos que se indiquen según sus identificadores.

- `commonfactor`. Para hallar el factor común de varios términos se puede utilizar la función de SymPy *together* [32]. El inconveniente de esta función es que dependiendo de las expresiones también las opera, y eso es algo que se quiere evitar para esta funcionalidad. Para solventarlo ha sido necesario modificar internamente *together*, y se le ha añadido la funcionalidad de indicar el factor a extraer.

- `delcommonfactor`. Cuando ambos miembros de la ecuación tienen la misma operación y alguno de sus términos coincide en ambas partes, se puede conseguir una ecuación equivalente eliminándolos. *delcommonfactor* da esa funcionalidad a la aplicación, comprobando si los términos indicados efectivamente pueden eliminarse de ambos miembros y eliminándolos en caso afirmativo.

Durante la iteración de las pruebas (ver capítulo 7) se desarrolló una específica relacionada con la resolución automática de las ecuaciones siguiendo los pasos mencionados en el Anexo C. Debido a su buena respuesta y su gran utilidad para la aplicación, se decidió incluir como parte de la funcionalidad del proyecto: *solve*.

6.3. Template

Los templates definen la interfaz gráfica; son la forma en la que Django genera los HTMLs dinámicamente [33]. Cada template contiene unas partes estáticas de código HTML a mostrar junto con una sintaxis especial que describe cómo quiere presentarse la información dinámica.

A continuación se va hablar sobre cuatro aspectos clave relacionados con la interfaz gráfica de la aplicación: archivos HTML, LaTeX, MathML y JavaScript.

6.3.1. HTML

A pesar de que Django tiene una gran variedad de templates predefinidos, ha sido necesario desarrollar nuevos templates que se adapten al uso que se quiere hacer de la aplicación implementados en lenguaje HTML con la notación definida por Django [34].

Cada template contiene unas variables que serán reemplazadas por el contenido que reciban de las vistas, y unas etiquetas que permiten controlar su lógica. A continuación se muestra un pequeño ejemplo con parte del código de *notebook.html*:

```
{\% for action in actions \%}
  <li><a href=‘‘{\% url ‘chalkboard’ equation.id \%}’’>{{
    ↪  equation }}</a>
    {\% if equation.solved \%}
      <img src=‘‘{\% static ‘chalkpy/images/true.png’ \%}
        ↪ ’’/>
    {\% endif \%}
  </li>
{\% endfor \%}
```

La aplicación está formada por cinco templates: *homepage.html*, *notebook.html*, *chalkboard.html*, *equation.html* y *steps.html*.

- `homepage`. Página de inicio de la aplicación, describe ChalkPy y sus funcionalidades.
- `notebook`. Muestra un listado en LaTeX de las ecuaciones disponibles para el usuario a través de la lista dinámica de ecuaciones que le manda la vista (ver subsección 6.3.2).
- `chalkboard`. Página principal de la aplicación. Engloba la parte estática; la parte dinámica se actualiza a través de JQuery a partir de otros templates (ver subsección 6.3.4).
- `equation`. Parte dinámica de la página principal que engloba la ecuación, que se muestra a través de MathML (ver subsección 6.3.3).

- `steps`. Muestra dinámicamente el registro de acciones que se han realizado sobre la ecuación en LaTeX.

6.3.2. LaTeX

LaTeX se ha utilizado en dos aspectos de la aplicación: para escribir las ecuaciones de la interfaz gráfica —a excepción de la ecuación sobre la que se realizan las acciones— y para generar el documento PDF que guarda los pasos realizados sobre la ecuación.

Para llevar a cabo el primer aspecto, SymPy tiene una función que permite obtener el código LaTeX de una expresión [35]. Se almacena como variable en la base de datos y se le da formato LaTeX en el HTML correspondiente. El uso de LaTeX para mostrar las ecuaciones facilita la lectura de las mismas respecto a una cadena simple de caracteres, especialmente las fracciones. Sin embargo, tiene un contrapunto y es que LaTeX escribe los términos con un orden predeterminado, por lo que varias veces no coincide la ecuaciones con la que se interactúa con la que aparece en el último paso del cuaderno.

Cuando se presentan las ecuaciones en LaTeX es necesario tener en cuenta aquellas que tienen incógnitas propias. Esto se refleja en las diferentes variables que tiene la clase `Action` relacionadas con su texto:

- `originaltext`. Ecuación que ha introducido el usuario, con los nombres de las incógnitas propias si se utilizan.

- `operatingtext`. Ecuación con las incógnitas propias adaptadas para trabajar con SymPy.

- `latex`. Ecuación en formato LaTeX con las incógnitas propias si se utilizan.

Se ha utilizado la biblioteca PyLaTeX (ver subsección 3.3) para especificar el contenido de los documentos PDF y generarlos con LaTeX.

6.3.3. MathML

MathML [12] es un lenguaje basado en XML para expresar contenido matemático en la web. Su elección para representar la ecuación dinámica se debe a que su formato marcado facilita la identificación de los términos de la ecuación con los que el usuario interactúa.

A continuación se muestra la ecuación utilizada como ejemplo anterior en MathML:

```
<math display=‘‘block’’ xmlns=‘‘http://www.w3.org/1998/
  ↪ Math/MathML’’>
  <mfrac>
    <mrow>
      <mi>2</mi>
      <mo>*</mo>
      <mi>x</mi>
      <mo>+</mo>
      <mi>2</mi>
    </mrow>
    <mrow>
      <mi>3</mi>
    </mrow>
  </mfrac>
  <mo>=</mo>
  <mrow>
    <mi>5</mi>
    <mo>-</mo>
    <mi>x</mi>
    <mo>-</mo>
    <mi>1</mi>
  </mrow>
</math>
```

Al igual que con LaTeX, SymPy tiene una función para obtener el código MathML de una expresión [36], pero lo genera sobre una versión desactualizada. Por ello ha sido necesario crear una función que genere este código MathML a partir de la ecuación de forma que los navegadores lo representen correctamente.

6.3.4. JavaScript

Para terminar de dar forma a los templates se hace uso de JavaScript. Sus funcionalidades principales son:

1. Manejar la funcionalidad Drag&Drop de la ecuación.

2. Manejar las acciones que realiza el usuario sobre la ecuación.

3. Enviar las acciones realizadas sobre la ecuación a la vista a través de AJAX, sin necesidad de recargar toda la página. Las partes que sí necesitan actualización son la ecuación y los pasos.

6.4. View

Las vistas responden a los eventos que realiza el usuario sobre la interfaz e invocan al modelo para obtener información cuando es necesaria, que después devuelven a los templates. Por lo tanto, son intermediarias entre modelo y templates.

Las vistas en Django son métodos agrupados en el archivo *views.py* [37] que reciben una petición web y devuelven una respuesta HTTP que contiene la información dinámica necesaria para generar el archivo HTML definido como template.

Las vistas más relevantes de la aplicación, estrechamente relacionadas con las funciones mencionadas anteriormente en la subsección 6.2.2, son:

- `addeq`. Recibe la ecuación que el usuario quiere añadir y comprueba si está definida en el ámbito de la aplicación. En caso afirmativo, genera los objetos necesarios en la base de datos para poder trabajar con ella.
- `loadeq`. Devuelve la información necesaria para generar la ecuación a través de MathML en el template correspondiente. En caso de que se haya llamado debido a una interacción en la interfaz por parte del usuario, comprueba si dicha acción es correcta y actúa en consecuencia.
- `reload, undo, redo, descriptioneq, solveeq, printeq`. Llamadas a través de los botones de la interfaz definidos en el apartado 5.3.1 modifican la base de datos según la acción.

Cuando el usuario realiza una petición a una URL de la aplicación —definidas en el archivo *urls.py*—, esta llama a su vista asociada. La vista procesa la petición, se comunica con la base de datos mediante ORM –que permite realizar consultas en Python en vez de en SQL— y responde con un objeto *HTTPResponse* o genera una excepción en caso de que haya ocurrido algún error. La respuesta suele contener el template a cargar y un objeto conocido como *context*, que es un diccionario que contiene la información que necesita el template. Esta secuencia es reflejo de la figura 5.5, introducida en la subsección 5.5.

7
Pruebas y resultados

Se han realizado diversos tipos de pruebas sobre ChalkPy con el fin de buscar sus errores. Estas pruebas se presentan a continuación según el ámbito de la aplicación en el que se han realizado, de forma muy ligada a las iteraciones mencionadas en la subsección 3.2. Tras ellas se presentan los resultados de la aplicación.

7.1. Pruebas sobre la lógica

Debido a la relevancia del desarrollo de la lógica de ChalkPy, durante las iteraciones 1-6 se ha verificado su funcionamiento exhaustivamente.

7.1.1. Verificación

La verificación consiste en comprobar que el desarrollo del software es correcto, es decir, que funciona bien la herramienta. Para ello se han realizado inspecciones de código, pruebas de caja blanca y pruebas de caja negra.

Inspección de código

La inspección de código consiste en revisar el código implementado. Esta revisión se ha llevado a cabo en cada iteración con el principal objetivo de supervisar la estructura del código desarrollado y su complejidad.

Durante este tipo de prueba se dedujo que también es conveniente repetir esta

inspección pasado un tiempo desde la creación del módulo, para poder así enfocar desde otro punto de vista sus funcionalidades, por lo que en cada iteración se inspeccionaba también el código generado en las anteriores.

Por ejemplo, durante la inspección de la segunda iteración —funcionalidad simplificar— el código de la primera iteración —mover— sufrió varios cambios debido a la nueva funcionalidad implementada, que incluyeron tanto una reestructuración del código como la eliminación de funciones obsoletas.

Pruebas de caja blanca

Las pruebas de caja blanca consisten en la revisión del funcionamiento interno del programa a partir de su lógica. Para ello, se ejecutan todos los posibles caminos que se pueden seguir en el código implementado, intentando provocar situaciones extremas.

Este tipo de prueba se realizó a través de un main específico para la lógica de la herramienta. Con él se han diseñado diferentes pruebas relacionadas con el camino básico que debería seguir el código y con condiciones límite para encontrar errores.

Tras la realización de estas pruebas de caja blanca se han corregido los errores encontrados, relacionados principalmente con situaciones límite, aunque también algunos relacionados con funciones condicionales.

Pruebas de caja negra

Las pruebas de caja negra consisten en comprobar el buen funcionamiento del software sin tener en cuenta el procedimiento que se ha llevado a cabo.

Debido a la gran variedad de situaciones que pueden darse en la lógica de ChalkPy, en la iteración 5 se crearon diferentes tipos de pruebas que cubriesen todos los casos lógicos posibles, teniendo en cuenta el posible error humano que se hubiese cometido en las pruebas anteriores de caja blanca.

Las pruebas son:

1. Modificar aleatoriamente la ecuación, comprobando que su solución no cambia.
2. Operar sobre la ecuación aleatoriamente hasta conseguir su solución, siendo esta proporcionada.
3. Obtener el mínimo de pasos tras realizar varias veces la prueba anterior.
4. Resolver la ecuación paso a paso según lo definido en el Anexo C.

A continuación se describen con más detalle.

1. Modificar aleatoriamente la ecuación

Esta prueba consiste en ejecutar acciones de forma aleatoria sobre una ecuación. Si la ecuación obtenida tras la acciones tuviese una solución diferente a la de la ecuación inicial (que se conoce a través de la función *solve* de SymPy), entonces necesariamente se ha producido un error en esas acciones.

Con ella se ha pretendido cubrir el mayor número posible de casos límite. Los errores que han destacado tras esta prueba estaban relacionados con multiplicaciones y divisiones y con elementos clave como ceros y unos. Por ejemplo, en un principio se permitía mover el cero solitario de un miembro de la ecuación al otro lado dividiendo, lo cual supone una expresión dividida por cero.

2. Obtener aleatoriamente la solución

Muy similar a la prueba anterior pero con una finalidad distinta, se pretende con ella conseguir resolver la ecuación tras las operaciones aleatorias.

Esta prueba no sólo cubre posibles casos límite, sino que además proporciona los pasos a realizar para resolver la ecuación. Sin embargo, se debe tener en cuenta que esos pasos, aunque resuelvan la ecuación, son aleatorios, por lo que mayoritariamente no corresponden con los pasos para una resolución eficiente.

Dependiendo de la complejidad de la ecuación, se consigue obtener la solución tras más o menos operaciones. Por ello, este tipo de prueba fue el más adecuado para ecuaciones simples. En el caso de las complejas, la ecuación tendía a crecer debido a la alta frecuencia de acciones de tipo *commonfactor* y no sólo se volvía poco práctica sino que el número de operaciones innecesarias era realmente alto. Por esto se realizó una adaptación de la prueba para que resolviese utilizando únicamente las operaciones de *mover* y *simplificar*, lo cual supuso mejores resultados pero limitó el su uso sobre ecuaciones sin divisiones.

3. Pasos mínimos tras resolver aleatoriamente la ecuación

Al aplicar la prueba anterior repetidas veces sobre una misma ecuación, se obtienen varios conjuntos de pasos posibles que permiten su resolución. Entre todos esos conjuntos se ha querido destacar aquel con el mínimo número de pasos, puesto que su resolución es la más eficiente respecto al número de operaciones.

El éxito de esta prueba también ha dependido de la complejidad de la ecuación a probar. Con ecuaciones sencillas sí se conseguía el mínimo número de pasos necesarios, pero las ecuaciones complejas continuaban proporcionando un alto número de acciones innecesarias.

Debido a que tanto esta prueba como las dos anteriores estaban basadas principalmente en aplicar fuerza bruta al programa, sin tener en cuenta el método de resolución de ecuaciones que se suelen utilizar en la realidad, se vio necesario realizar la prueba siguiente.

4. Método de resolución paso a paso

La prueba de resolución de ecuaciones paso a paso no sólo permitió comprobar el funcionamiento de la herramienta sino que también añadió una funcionalidad extra a la aplicación que no se había tenido en cuenta hasta ese momento: obtener la solución de la ecuación y los pasos para obtenerla de forma automática.

7.2. Pruebas sobre la interfaz

También se ha tenido especial cuidado durante las pruebas de verificación y validación de la interfaz, pues se quiere mostrar al usuario una aplicación usable y funcional.

La verificación de la interfaz de usuario de la aplicación se ha centrado en la comprobación y corrección de errores en las iteraciones 7 y 8 a través de pruebas de caja negra y caja blanca.

7.2.1. Validación

Las pruebas de validación permiten comprobar que la aplicación que se está creando cumple con los requisitos del usuario, por lo que son poco eficaces en un desarrollo con metodología incremental ya que esas funcionalidades van desarrollándose poco a poco. Por ello, este tipo de pruebas se han dejado para la última iteración del desarrollo de la aplicación.

En todas las pruebas descritas a continuación no solo se ha interactuado con el programa a través del ordenador sino también a través de una pizarra digital para conocer así la usabilidad sobre dicho sistema, ya que es una tecnología que comienza a estar presente en las aulas.

Prueba de aceptación

La prueba de aceptación está enfocada al cliente y consiste en la comprobación de que se cumplen todos los requisitos que solicitó. En el caso de esta aplicación esa función ha sido ejercida por el tutor.

Prueba alfa

Para la realización de pruebas alfa se ha contado con la participación de usuarios de distintos perfiles para que probasen la aplicación, diesen su opinión y reportaran errores.

La opinión general sobre la aplicación ha sido positiva respecto a su funcionalidad y su usabilidad. Los usuarios han reportado errores y también han aportado nuevas ideas

Pregunta	Promedio
¿Le parece ChalkPy un programa adecuado para introducir los conceptos básicos de las ecuaciones algebraicas?	4,62
¿Como profesor, utilizaría ChalkPy en su clase?	4,38
¿Considera ChalkPy útil para el estudio individual de los estudiantes?	4,23
¿Te resulta fácil de usar ChalkPy?	4,69

Tabla 7.1: Promedios de la encuesta realizada a los alumnos del Máster en Formación de Profesorado sobre ChalkPy

para la mejora de la misma. En el Anexo E se recogen algunos de sus comentarios.

Prueba beta

La prueba beta se ha enfocado al conjunto de usuarios a los que está dirigida esta aplicación: docentes y estudiantes.

Los betatesters con perfil de docentes que han realizado la prueba han sido los estudiantes y docentes de la Universidad Autónoma de Madrid que atienden e imparten, respectivamente, el Máster Universitario en Formación de Profesorado de Educación Secundaria Obligatoria y Bachillerato de la rama de Matemáticas durante la clase *Innovación docente e iniciación a la investigación educativa.*

En esta prueba se les explicó la motivación de la aplicación (ver apartado 1.1), se les mostró su uso y se les animó a probarla en una pizarra digital.

El objetivo principal de la prueba era conocer si tanto los docentes actuales como los futuros (los alumnos) están interesados en el uso de nuevas tecnologías en la educación y obtener su opinión sobre la aplicación. Como se ve en la tabla 7.1, los usuarios mostraron un interés medio-alto por la aplicación y su opinión general fue que la aplicación debería seguir desarrollándose para cubrir más aspectos del álgebra.

También se probó la aplicación con varios estudiantes de la ESO para conocer su opinión respecto a trabajar con la herramienta en clase. La funcionalidad que más les gustó fue que se pudiese resolver automáticamente.

En el Anexo E se recogen los resultados de ambas pruebas.

7.3. Resultados

Tras todo el desarrollo de este Trabajo de Fin de Grado se ha obtenido como resultado la herramienta ChalkPy, que cumple con todas las funcionalidades iniciales propuestas.

Con ella se solventan los aspectos negativos de la pizarra tradicional (ver capítulo 1.1) de la siguiente forma:

- No se pueden cometer errores durante la resolución de las ecuaciones.
- Los pasos realizados sobre la ecuación se muestran en forma de lista clara y descriptiva.
- El estudiante no necesita tomar apuntes durante la explicación, porque puede generar un documento que refleje los pasos realizados.
- El tiempo de respuesta de la aplicación no rompe el flujo natural de las explicaciones del docente.

Las funcionalidades desarrolladas —mover, simplificar, sacar factor común y eliminarlo— han resultado suficientes para la resolución de las ecuaciones de forma práctica.

La página principal de ChalkPy (ver la figura 7.1) contiene toda la información relevante para la resolución de la ecuación de forma concisa y sencilla.

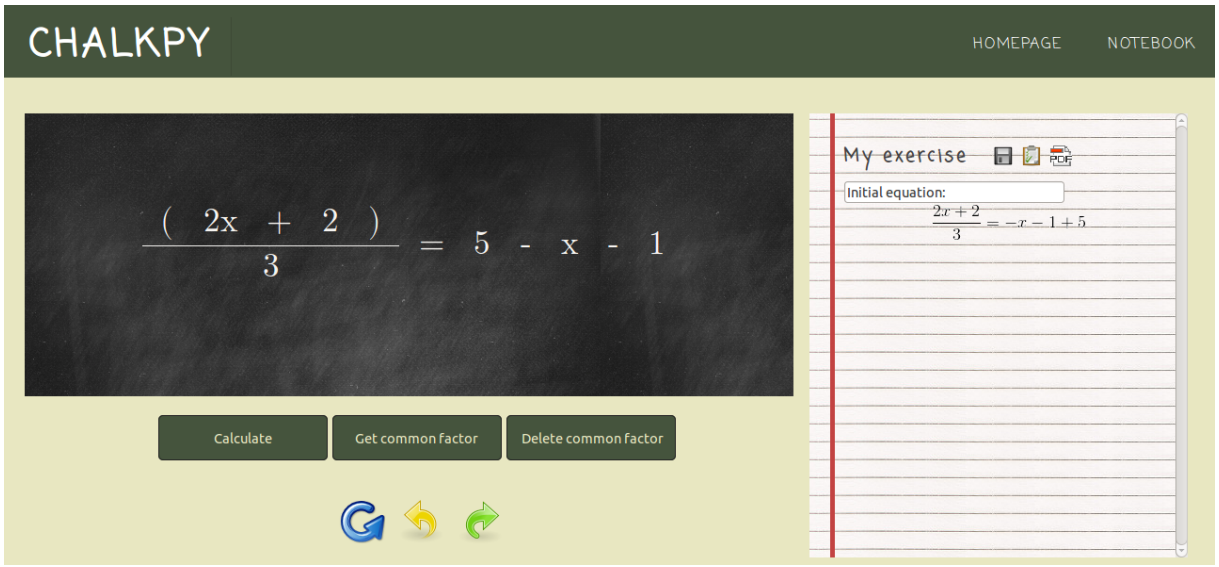

Figura 7.1: Página chalkboard de ChalkPy

Por último, los docentes que han realizado las pruebas han expresado interés en utilizar la aplicación durante sus clases, principalmente para evitar a sus alumnos copiar apuntes y mejorar su atención. Ven la herramienta como algo útil para las explicaciones y la corrección de ejercicios por parte del estudiante tanto en el aula como en clase. Esto coincide con la opinión de los estudiantes, que creen que la aplicación es útil para las horas de clase pero no para hacer deberes, pues no practicarían el cálculo —sumar, restar, etc.— . Tanto docentes como estudiantes se han sentido satisfechos con ChalkPy y muestran un alto interés en que se continúe su desarrollo, ampliando el alcance de resolución de ecuaciones.

8

Conclusiones y líneas de trabajo futuro

Tras el desarrollo de este Trabajo de Fin de Grado se ha hecho de ChalkPy una herramienta que cumple su objetivo: permite el desarrollo dinámico de ecuaciones polinómicas de primer grado con solución única para utilizarlo en un ambiente educativo a través de su proyección en el aula desde un ordenador.

ChalkPy reúne el dinamismo de aplicaciones como Mathination y Algebra Touch para ponerlo a disposición de los usuarios como aplicación web de código abierto para promover el uso de tecnologías en el aula.

El desarrollo del proyecto ha seguido un ciclo de vida iterativo incremental con ocho iteraciones según la funcionalidad a implementar: mover, simplificar, sacar factor común, eliminar factor común, pruebas, LaTeX, interfaz web dinámica y otras funcionalidades. Actualmente se pretende comenzar otra iteración con la que hacer la aplicación accesible a los usuarios.

ChalkPy se ha construido en Python y con Django como framework para desarrollar la aplicación web, que sigue un patrón de arquitectura Model-View-Template (MTV). En ella el componente *modelo* contiene la información de la aplicación en una base de datos en SQLite. Los *template* definen la interfaz gráfica generando HTMLs dinámicamente según la información que quiere mostrarse por pantalla, donde destaca el uso de LaTeX y MathML para la representación de las ecuaciones. Las *vistas* manejan la navegación en la aplicación y sirven como intermediarias entre el modelo y los templates.

El éxito del proyecto se ha consolidado durante la prueba realizada a los alumnos del Máster Universitario en Formación de Profesorado de Educación Secundaria Obligatoria y Bachillerato de la rama de Matemáticas, en donde se introdujo ChalkPy como herramienta de innovación docente y obtuvo una buena crítica.

Con todo esto, se tiene especial interés en continuar el desarrollo de ChalkPy.

Lo primero será revisar las recomendaciones hechas por los usuarios durante la fase de pruebas, entre las que destacan alternativas de interacción con la ecuación y añadir un componente de gamificación a la aplicación para promover su uso entre los estudiantes.

Las pruebas han demostrado que los docentes con acceso a pizarras digitales priorizarían su uso sobre el ordenador, por lo que se pretende adaptar el contenido a las posibilidades de la pizarra digital.

Y, por último, lo más solicitado ha sido la ampliación de su funcionalidad para que cubra más aspectos del álgebra. El siguiente paso es la inclusión de operaciones con potencias y raíces.

Referencias

Las siguientes direcciones web enlazan a información extendida sobre las herramientas, lenguajes de programación y librerías mencionadas en la memoria.

[1] Eduard 1845-1930 Muhammad ibn Ahmad 973?-1048; Sachau. *Alberuni's India. An account of the religion, philosophy, literature, geography, chronology, astronomy, customs, laws and astrology of India about A.D. 1030*. 1910, pág. 182. URL: https://archive.org/details/alberunisindiaac01biru.

[2] *Mathway*. [Último acceso: 18/01/2016]. URL: https://mathway.com/.

[3] *Mathination*. [Último acceso: 18/01/2016]. URL: http://www.mathination.com/.

[4] *Algebra Touch*. [Último acceso: 18/01/2016]. URL: http://www.regularberry.com/.

[5] Ian Sommerville. *Ingeniería del Software. Séptima edición*. Pearson Educación, 2005, págs. 59-79. ISBN: 84-7829-074-5. URL: http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/ngenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf.

[6] *Acceso consola SymPy*. [Último acceso: 18/01/2016]. URL: http://live.sympy.org/.

[7] *Python*. [Último acceso: 18/01/2016]. URL: https://www.python.org/.

[8] *SymPy*. [Último acceso: 18/01/2016]. URL: http://www.sympy.org/es/.

[9] *Django*. [Último acceso: 18/01/2016]. URL: https://www.djangoproject.com/.

[10] *SQLite*. [Último acceso: 18/01/2016]. URL: https://www.sqlite.org/.

[11] *LaTeX*. [Último acceso: 18/01/2016]. URL: https://www.latex-project.org/.

[12] *Biblioteca MathML*. [Último acceso: 18/01/2016]. URL: https://www.w3.org/Math/.

[13] *HTML*. [Último acceso: 18/01/2016]. URL: https://es.wikipedia.org/wiki/HTML.

[14] *JavaScript*. [Último acceso: 18/01/2016]. URL: https://es.wikipedia.org/wiki/JavaScript.

[15] *JQuery*. [Último acceso: 18/01/2016]. URL: https://jquery.com/.

[16] *Sublime Text*. [Último acceso: 18/01/2016]. URL: http://www.sublimetext.com/.

[17] *PyCharm*. [Último acceso: 18/01/2016]. URL: https://www.jetbrains.com/pycharm/.

[18] *PyLaTeX*. [Último acceso: 18/01/2016]. URL: https://github.com/JelteF/PyLaTeX.

[19] *GitHub*. [Último acceso: 18/01/2016]. URL: https://github.com/.

[20] *Dia*. [Último acceso: 18/01/2016]. URL: http://dia-installer.de/index.html.es.

[21] *Balsamiq*. [Último acceso: 18/01/2016]. URL: https://balsamiq.com/.

[22] *Cacoo*. [Último acceso: 18/01/2016]. URL: https://cacoo.com/.

[23] *Gimp*. [Último acceso: 18/01/2016]. URL: https://www.gimp.org/.

[24] *Modelo-Vista-Controlador*. [Último acceso: 18/01/2016]. URL: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[25] *Model-Template-View*. [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/en/1.9/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names.

[26] *Django Image*. [Último acceso: 18/01/2016]. URL: http://blog.easylearning.guru/wordpress/wp-content/uploads/2015/08/Django-Template.png.

[27] *Estructura de Django*. [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/en/1.9/intro/tutorial01/.

[28] *Explicación del modelo de Django*. [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/en/1.9/intro/tutorial02/.

[29] *Explicación de la clase Equality de SymPy*. [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/latest/modules/core.html?highlight=eq#sympy.core.relational.Equality.

[30] *Explicación de la manipulación de expresiones matemáticas a través de árboles*. [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/dev/tutorial/manipulation.html.

[31] *Descripción de la función simplify de SymPy*. [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/dev/gotchas.html?highlight=simplify#sympy.simplify.simplify.

[32] *Descripción de la función together de SymPy*. [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/dev/modules/polys/reference.html?highlight=together#sympy.polys.rationaltools.together.

[33] *Explicación de los templates de Django*. [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/en/1.9/topics/templates/.

[34] *Explicación del lenguaje a utilizar en los templates de Django*. [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/en/1.9/ref/templates/language/.

[35] *Descripción de la función latex de SymPy.* [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/dev/modules/galgebra/printing.html?highlight=latex#sympy.galgebra.printing.latex.

[36] *Descripción de la función latex de SymPy.* [Último acceso: 18/01/2016]. URL: http://docs.sympy.org/latest/modules/printing.html?highlight=mathml#sympy.printing.mathml.mathml.

[37] *Explicación de las views de Django.* [Último acceso: 18/01/2016]. URL: https://docs.djangoproject.com/es/1.9/topics/http/views/.

[38] *Conceptual Math.* [Último acceso: 18/01/2016]. URL: http://www.conceptuamath.com.

[39] *Wolfram Alpha.* [Último acceso: 18/01/2016]. URL: http://www.wolframalpha.com/.

[40] *Geogebra.* [Último acceso: 18/01/2016]. URL: https://www.geogebra.org/.

[41] *Calculadora gráfica Mathlab.* [Último acceso: 18/01/2016]. URL: https://play.google.com/store/apps/details?id=us.mathlab.android&hl=es.

[42] *Intermatia.* [Último acceso: 18/01/2016]. URL: https://www.intermatia.com.

[43] *DragonBox Álgebra 12+.* [Último acceso: 18/01/2016]. URL: http://www.dragonboxapp.com/.

[44] *Photomath.* [Último acceso: 18/01/2016]. URL: https://play.google.com/store/apps/details?id=com.microblink.photomath&hl=es.

[45] *Mathspace.* [Último acceso: 18/01/2016]. URL: https://mathspace.co/.

[46] *Microsoft Mathematics.* [Último acceso: 18/01/2016]. URL: https://www.microsoft.com/es-es/download/details.aspx?id=15702.

[47] *Math Solver.* [Último acceso: 18/01/2016]. URL: https://itunes.apple.com/es/app/fx-math-solver/id493941470?mt=8.

[48] *Cymath.* [Último acceso: 18/01/2016]. URL: http://www.cymath.com/.

[49] José Colera et al. *Matemáticas 3.* Editorial Anaya, 2006. ISBN: 84-667-1039-6.

[50] *Imágenes para fondo pizarra.* [Último acceso: 19/01/2016]. URL: http://designpieces.com/2014/02/chalkboard-look-and-feel/.

[51] *Imagen cuaderno a rayas.* [Último acceso: 19/01/2016]. URL: https://www.flickr.com/photos/creative_stock/17375986142.

[52] *Template en el que se basa homepage.* [Último acceso: 19/01/2016]. URL: http://templated.co/ion.

[53] *Template en el que se basan notebook y chalkboard.* [Último acceso: 19/01/2016]. URL: http://templated.co/entryway.

[54] *Página de templates.* [Último acceso: 19/01/2016]. URL: http://templated.co/.

[55] *Imagen alumno resolviendo una ecuación en la pizarra.* [Último acceso: 19/01/2016]. URL: https://www.flickr.com/photos/simpleinsomnia/11125348744/.

[56] *Imagen check verde*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/d/d9/Green_check.png.

[57] *Imagen delete*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/2/21/DeleteWinIcon.png.

[58] *Imagen add*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/0/06/Farm-Fresh_add.png.

[59] *Imagen undo*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/9/92/Edit-undo.svg/48px-Edit-undo.svg.png.

[60] *Imagen redo*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c1/Gnome-edit-redo.svg/48px-Gnome-edit-redo.svg.png.

[61] *Imagen reload*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/b/b0/Reload_all_tabs.png.

[62] *Imagen save*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/f/f6/Toolbar_save_changes.png.

[63] *Imagen solve*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/0/03/Edit-check-sheet.svg/48px-Edit-check-sheet.svg.png.

[64] *Imagen pdf*. [Último acceso: 19/01/2016]. URL: https://upload.wikimedia.org/wikipedia/commons/8/84/Icon_pdf.gif.

Apéndices

A

Manual de Usuario

Al ejecutar la aplicación se muestra una pantalla de bienvenida en la que se da acceso a la aplicación y se muestran sus características (ver figura A.1). Haciendo clic en "Let's start" se accede a la página denominada "Notebook" (*cuaderno*).

Figura A.1: Bienvenida a ChalkPy

En la página Notebook (A.2) se muestra una lista de ecuaciones. Para añadir una nueva ecuación, se hace clic en (+). Aparecerá un prompt donde introducir el texto. En el caso de querer utilizar variables propias es necesario ponerlas entrecomilladas, como muestra la figura A.3. Haciendo clic en Aceptar aparecerá una nueva ecuación en nuestra

lista figura A.4).

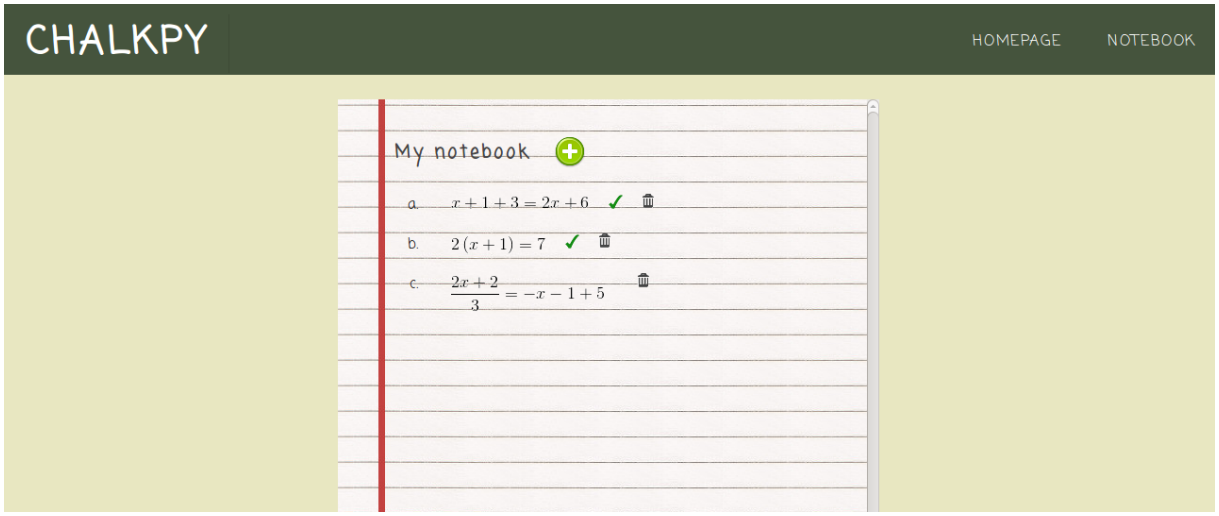

Figura A.2: Bienvenida a ChalkPy

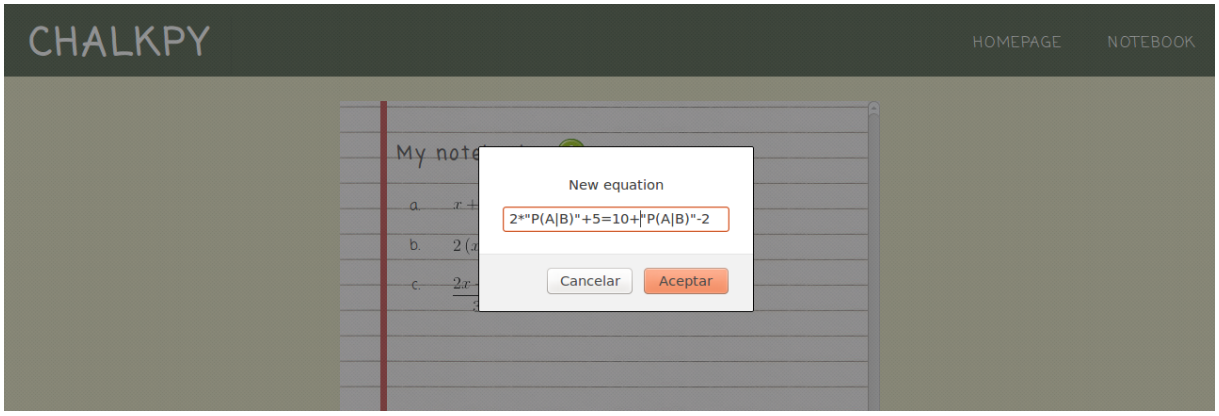

Figura A.3: Página Notebook

El check verde indica que la ecuación ha sido resuelta, y a través del botón con icono de papelera se puede eliminar. Si se hace clic en cualquiera de las ecuaciones se accede a la página denominada "Chalkboard" (*pizarra*), donde se puede interactuar con la ecuación seleccionada.

Chalkboard A.5 se divide visualmente en dos áreas que contienen diferentes elementos enfocados a la funcionalidad de cada una: la ecuación y el cuaderno de pasos.

Para mover un término por la ecuación se utiliza el método Drag&Drop, para lo cual es necesario hacer clic con el botón izquierdo del ratón sobre el término y, sin soltar, moverlo hacia la posición deseada (ver figura A.6).

Bajo la ecuación se reúnen los elementos a través de los cuales interactuar:

- El botón "Calculate" permite operar los términos seleccionados mediante doble clic. En la figura A.7 se quiere realizar la multiplicación del miembro derecho, viéndose su resultado en la figura A.8.

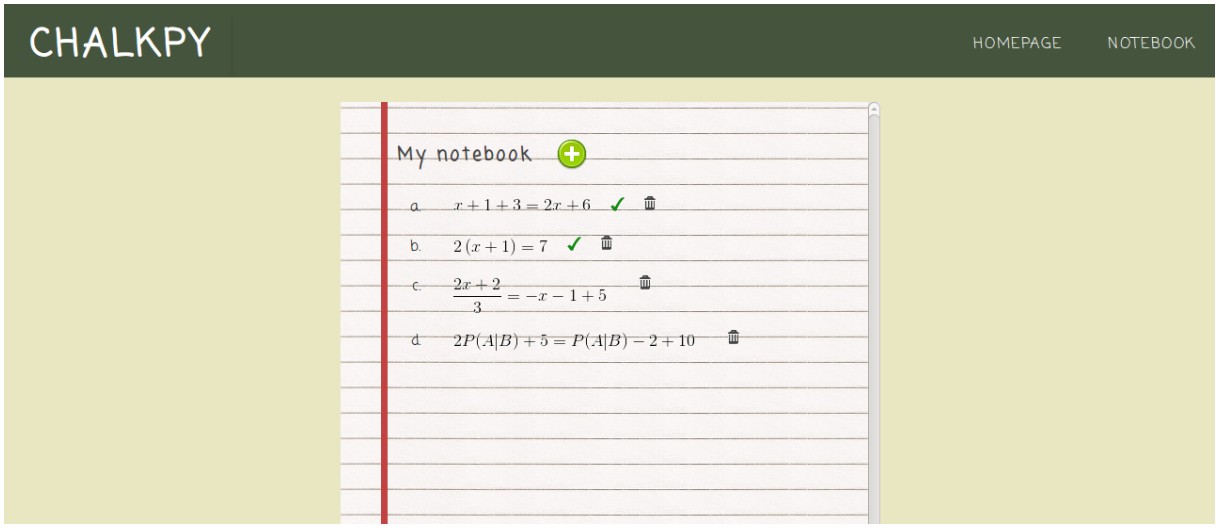

Figura A.4: Agregando una ecuación al cuaderno

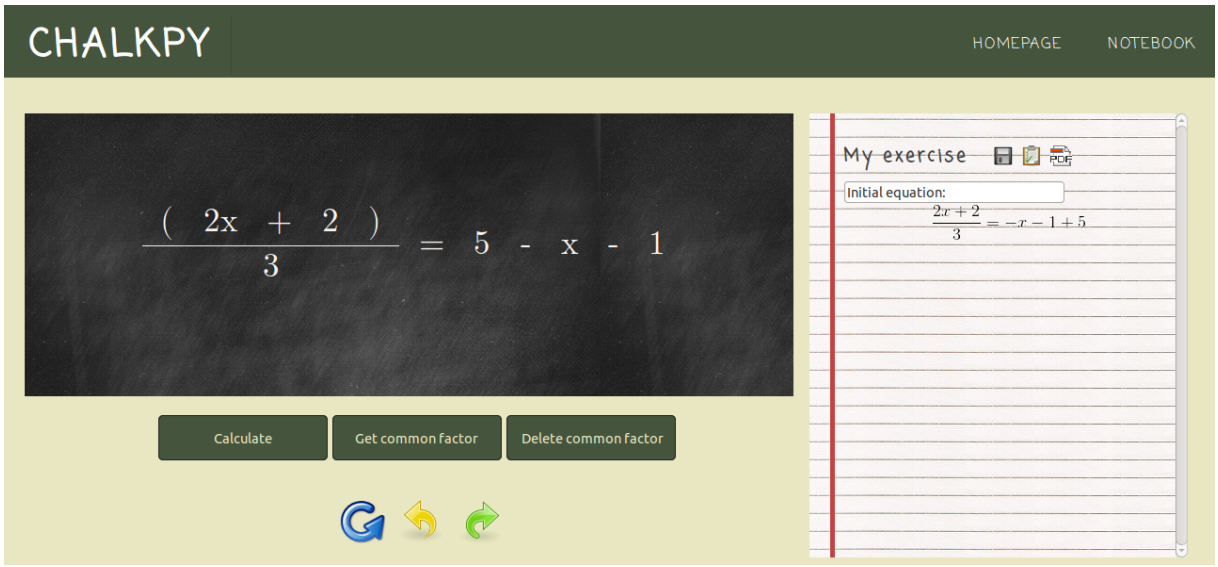

Figura A.5: Página Chalkboard

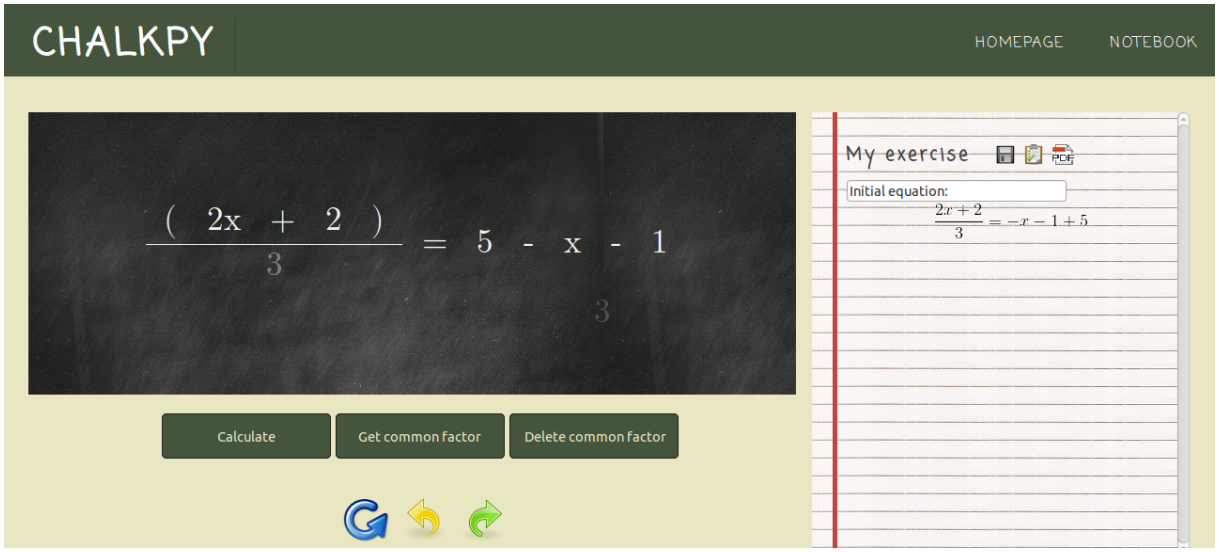

Figura A.6: Mover término de la ecuación

- El botón "Common factor" saca factor común de los términos seleccionados mediante doble clic.
- El botón "Delete common factor" elimina los términos seleccionados mediante doble clic si son comunes en ambos miembros.
- La flecha azul permite reiniciar la ecuación.
- La flecha amarilla permite volver a pasos anteriores.
- La flecha verde permite regresar a pasos posteriores que se hayan realizado.

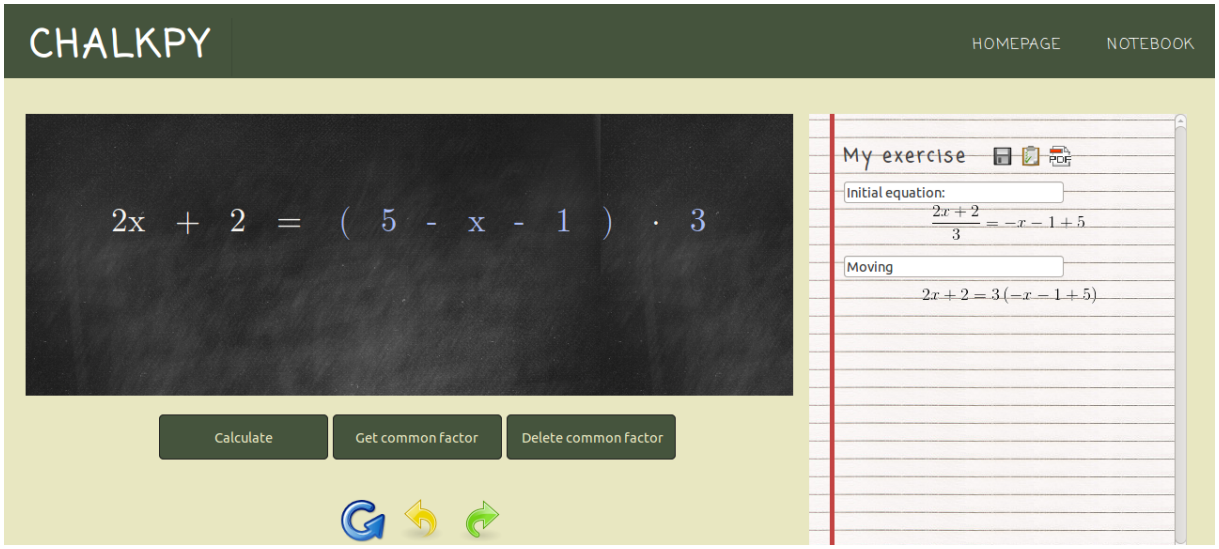

Figura A.7: Seleccionando términos para aplicar acción operar, sacar factor común o eliminarlo

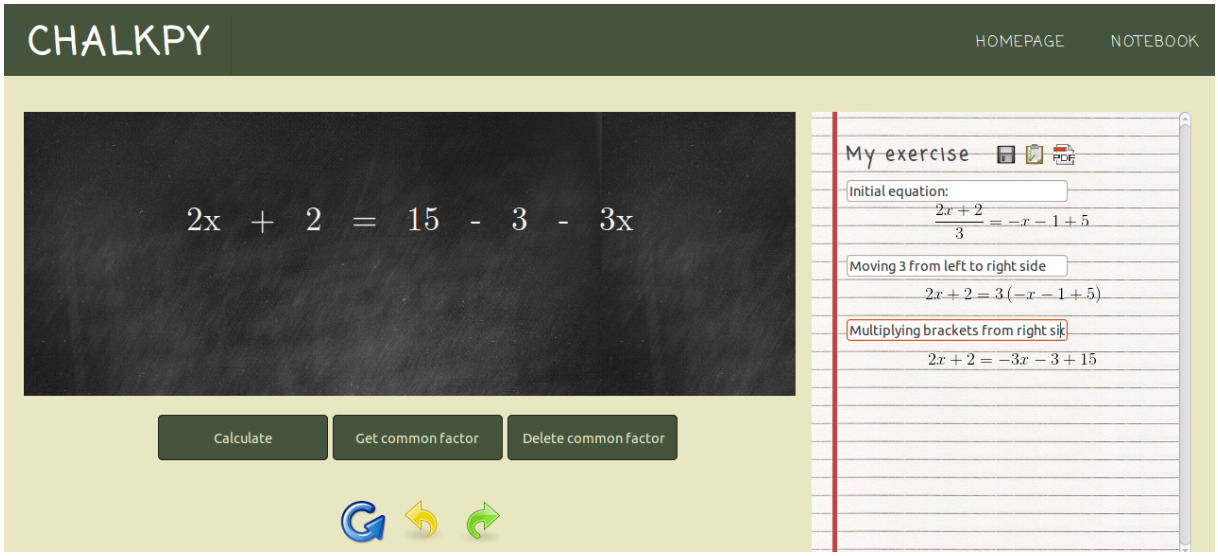

Figura A.8: Resultado de calcular la multiplicación del miembro derecho

La parte B, en la parte derecha de la pantalla, muestra un listado de las acciones que se han realizado sobre la ecuación. El texto de cada acción se puede modificar para tener descripciones más completas. En caso de modificarlo, dichos cambios se guardan pulsando el icono del disquete, tras lo cual le aparecerá un mensaje confirmando que se han guardado correctamente (ver figura A.9).

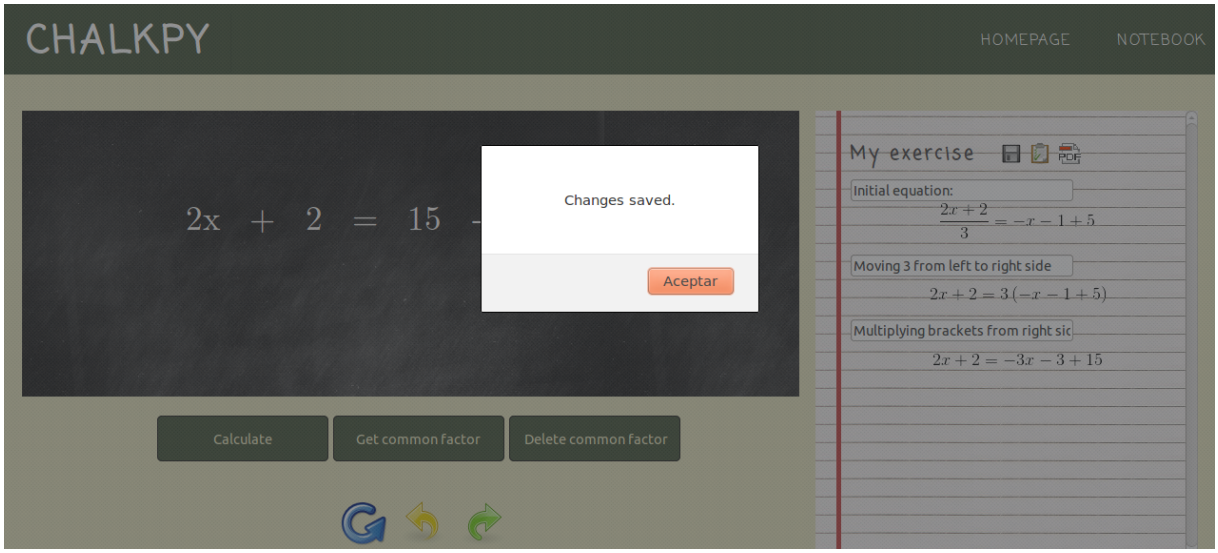

Figura A.9: Confirmación de que se han guardado las descripciones

El botón a la derecha del disquete permite resolver la ecuación de forma automática, por lo que al pulsarle aparecerá una lista de todas las acciones que ha sido necesario realizar, como se ve en la figura A.10.

Para guardar todos los pasos realizados en un documento PDF, se hace clic en el botón correspondiente. Tras ello aparecerá un mensaje de aviso para indicar que el archivo se

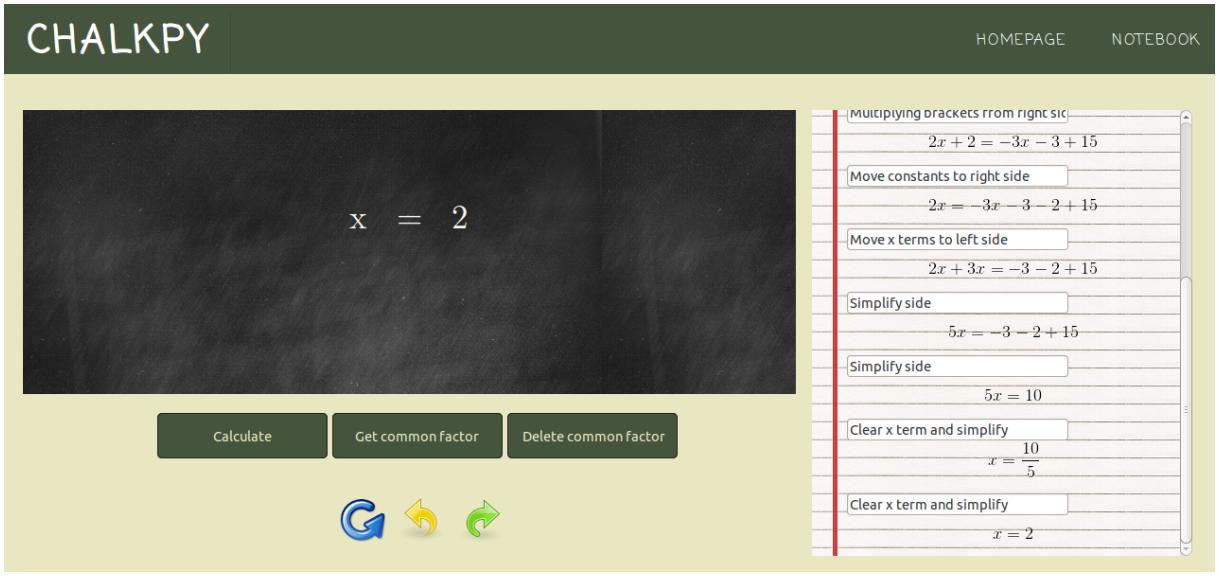

Figura A.10: Ecuación resuelta de forma automática

ha guardado (ver figura A.11).

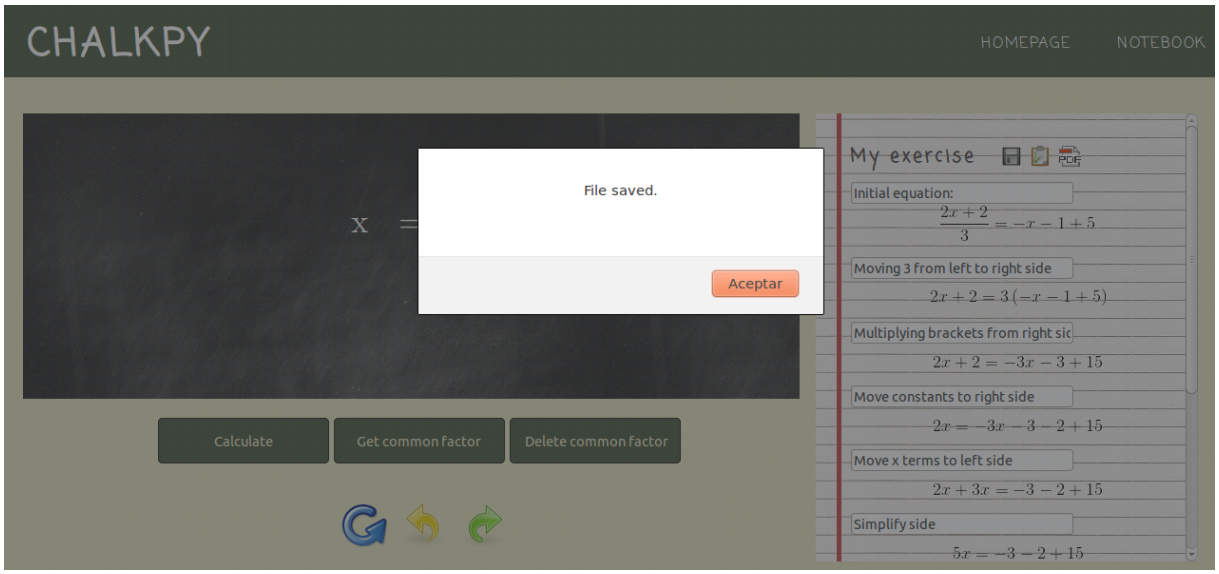

Figura A.11: Confirmación de archivo guardado

B

Otras herramientas matemáticas

A continuación se mencionan varias herramientas y aplicaciones móviles relacionadas con las matemáticas, cuyo uso puede darse tanto por parte del docente como del estudiante.

El área donde se encuentra más variedad de estas herramientas es el enfocado a estudiantes de primaria.

- *Conceptual Math* [38]. Enfoca la enseñanza de dos módulos —fracciones y multiplicaciones y divisiones— en el aula de forma visual y conceptual. El docente utiliza la herramienta para sus explicaciones y los estudiantes para realizar ejercicios durante las clases.
- *Wolfram Alpha* [39]. Motor de respuestas que ofrece, entre sus muchas características, la posibilidad de introducir ecuaciones y ver paso a paso el proceso a través del cual se obtiene su solución.
- *Geogebra* [40]. Con esta calculadora gráfica es posible resolver ecuaciones a través de su vista CAS (*Computer Algebra System*). Además, permite dibujarlas como funciones en el plano.
- *Calculadora gráfica Mathlab* [41]. Calculadora gráfica para Android que permite tanto dibujar ecuaciones en el plano como ver su resolución paso a paso.
- *Intermatia* [42]. Página web que ofrece ejercicios de matemáticas adecuados a ESO y Bachillerato. El estudiante introduce la solución a un ejercicio y, en caso de que sea incorrecta, la web le muestra el proceso paso a paso para resolverlo.

Las siguientes herramientas se centran principalmente en el álgebra elemental:

- *DragonBox Álgebra 12+* [43]. Este juego para dispositivos móviles está enfocado a aquellos estudiantes que tienen contacto con el álgebra por primera vez. En un principio, sin utilizar letras y números, sino a partir de una caja y varias imágenes, se introducen conceptos como "despejar la incógnita" o "trasponer términos". A medida que se avanza en el juego se va transitando al álgebra con letras y números. Está disponible para varias plataformas pero tiene un coste asociado.
- *Photomath* [44]. El principal atractivo de esta aplicación móvil reside en que el estudiante puede fotografiar una ecuación y la aplicación se encarga de resolverla paso a paso.
- *Mathspace* [45]. Aplicación para dispositivos móviles que permite al estudiante resolver ecuaciones. Su componente educativa reside en que comprueba que los pasos que el estudiante va siguiendo para resolver la ecuación son correctos y le proporciona pistas sobre cómo debería seguir operando en caso de que duda.
- *Microsoft Mathematics* [46]. Aplicación de escritorio que permite introducir ecuaciones para obtener su solución y ver los pasos que se han seguido hasta ella.
- *FX Math Solver* [47]. Aplicación para dispositivos móviles en la que, tras introducir una ecuación, te indica su resolución paso a paso.
- *Cymath* [48]. Aplicación web y móvil que permite resolver ecuaciones paso a paso.

C

Análisis de ecuaciones

A continuación se muestra un resumen del documento obtenido durante el análisis de requisitos sobre cómo resolver ecuaciones y cómo aplicarlo a la aplicación. Su contenido no coincide con la estructura final del proyecto.

Para resolver una ecuación polinómica de primer grado se recomienda seguir los siguientes pasos [49]:

1. Quitar denominadores. Esto se consigue transformando la ecuación a otra equivalente en la que todos los términos estén en forma de fracción y tengan un denominador común. Pasos a seguir:
   a) Calcular el mínimo común múltiplo (m.c.m) de todos los términos que aparecen en la ecuación.
   b) Calcular la fracción equivalente de cada término de la ecuación con el m.c.m obtenido en el paso anterior como denominador. Para ello:
      1) Dividir el m.c.m. entre el denominador original del término.
      2) Multiplicar el resultado con el numerador original.
2. Quitar paréntesis. Esto incluye tanto realizar las operaciones contenidas en el paréntesis, por prioridad, como realizar las operaciones que afectan los paréntesis.
3. Despejar los términos de grado uno (aquellos con x) a un miembro de la ecuación y los términos independientes al otro.
4. Simplificar cada miembro. Esto supone sumar los términos de cada miembro entre sí.

5. Despejar x. Tras esto se obtiene la solución de la ecuación, que debe ser siempre un valor irreducible.

6. Comprobación. Para asegurarnos de que no se ha cometido algún error durante la resolución de la ecuación, se sustituye la solución obtenida en las incógnitas de la ecuación inicial. Si ambos miembros muestran el mismo valor, no se ha cometido error alguno.

Con este conocimiento base y con la experiencia personal del autor de este documento, se han realizado una serie de ecuaciones (ejercicios del libro mencionado con anterioridad) para comprobar si esas son las operaciones básicas necesarias para resolver cualquier ecuación polinómica de primer grado con solución.

Relacionando las ecuaciones con el proyecto a realizar, encontramos que aparecerán con diferentes tipos de representación:

1. Variable string: cuando se introduzcan las ecuaciones por teclado.

2. Variable SymPy: para operar con ella.

3. Árbol con identificadores: conexión entre 2 y 4.

4. Gráficos: botones.

Es necesario tener en cuenta que habrá acciones que sólo afecten a la representación gráfica de la ecuación y otras que afecten a la ecuación en sí, por eso la individualidad de los puntos 2 y 3.

Se ha concluido que las operaciones básicas necesarias son las siguientes:

- Mover término por el miembro.
  - Nombre función: movein.
  - Funcionalidad: cambia la posición del término en su miembro.
  - Argumentos: identificador del término a mover y nueva posición.
  - Requisitos: sólo puede moverse en el mismo nivel que se encuentra. Los términos bajo la operación potencia no pueden moverse, pues su orden es relevante.
  - Funcionamiento: cambia el contenido de las ramas. Sólo afecta a la representación de la ecuación. Ojo porque no es un intercambio de posiciones, sino un desplazamiento.
  - Retorno: true o false.
- Mover término al otro miembro.
  - Nombre función: moveout
  - Funcionalidad: mueve un término al otro miembro de la ecuación.

- Argumentos: identificador del término a mover y nueva posición.
- Requisitos: sólo puede moverse si su nivel es menor a 2. Los términos bajo la operación potencia no pueden moverse, pues su orden es relevante.
- Funcionamiento: cambia el contenido de las ramas. Afecta tanto a la ecuación como a su representación.
- Retorno: true o false.

- Simplificar. Según los términos algebraicos, hace referencia a obtener una expresión equivalente. Por lo tanto, abarca tanto operar con términos (incluidos paréntesis) como reducir fracciones.
  - Nombre función: simplify.
  - Funcionalidad: aplica función de los términos a los términos.
  - Argumentos: términos a simplificar.
  - Requisitos: los términos están en el mismo nivel.
  - Funcionamiento: simplifica los términos. Afecta tanto a la ecuación como a su representación. Es necesario crear una nueva ecuación que contenga los cambios modificados. Hay dos posibles casos:
    - Caso ideal: operar todos los términos bajo el efecto de la función.
      - Ejemplo: `(6x+2+x)/3=(4-x)/3`. Se quiere sumar `(6x+2+x)`.
      - Qué hacer: aplicar la función que relaciona los términos. `eq.args[0].args[0].func(*eq.args[0].args[0].args)`
    - Caso probable: operar sólo un par de términos bajo el efecto de la función.
      - Ejemplo: `(6x+2+x)/3=(4-x)/3`. Se quiere sumar `(6x+x)`.
      - Qué hacer: identificar ambos términos y sumarlos bajo la forma: `eq.args[0].args[0].func(eq.args[0].args[0].args[0], eq.args[0].args`
    - Atención: Reducir fracciones o simplificar paréntesis se engloban en estos dos casos.
  - Retorno: true o false.

- Factor común.
  - Nombre función: commonfactor.
  - Funcionalidad: saca factor común de los términos indicados.
  - Argumentos: términos de los que sacar factor común.
  - Requisitos: los términos están en el mismo nivel.
  - Funcionamiento: obtiene factor común de los términos y crea una nueva multiplicación entre el factor común y lo restante de cada término. Afecta tanto a la ecuación como a su representación. Hay dos posibles casos:
    - Caso común: sacar factor común de todos los términos bajo el efecto de la función.
      - Ejemplo: `6x+2+8y=4-x`. factor común de `6x+2+8y`.

        - ◇ Qué hacer: identificar su ubicación y aplicar:
          together(eq.args[0]) —> `2(3x+1+4y)`
        - ◇ Ejemplo 2: `2x+2/3=4-x`. factor común de toda la ecuación.
        - ◇ Qué hacer: together(eq)
      - Caso posible: sacar factor común de sólo un par de términos. Misma conclusión que en el punto anterior.
    - Retorno: true o false.

- Quitar factor común.
  - Nombre función: delcommonfactor.
  - Funcionalidad: elimina el factor común de ambos miembros de la ecuación.
  - Argumentos: ecuación. Comprobar que es ecuación.
  - Funcionamiento. Hay dos posibles casos:
    - Caso ideal: la operación principal de cada miembro es la división.
      - ◇ Ejemplo: `(6x+2)/3 = (4-x)/3`
      - ◇ Qué hacer: crear nueva ecuación con la estructura que nos interesa.
        Datos:
        eq —> `(6x+2)/3=(4-x)/3`
        `eq.args[0].args[0] --> 6x+2`
        `eq.args[1].args[0] --> 4-x`
        Operaciones:
        ```
        eqleft = eq.args[0].args[0]
        eqright = eq.args[1].args[0]
        eqnew = Eq(eqleft, eqright)
        ```
    - Caso probable: la operación principal de cada miembro es la suma.
      - ◇ Ejemplo: `6x/3 + 2/3 = 4/3 - x/3`
      - ◇ Qué hacer: sacar factor común en cada miembro y aplicar el caso ideal.
    - Reflexión: Esta función también sirve para eliminar algo común en ambos miembros. Por ejemplo: `2+x+1=2+3x+2 --> x+1=3x+2`
  - Retorno: true o false.

Tras el análisis de cómo solucionar las operaciones básicas y las funcionalidades que pueden necesitarse, se concluye que se pueden resolver a partir de las siguiente cuatro funciones:

- Mover: move. Mueve los términos tanto internamente como externamente. Terminal: "`m0.0.0-0.0.2`".
- Simplificar: mysimplify. Opera. Terminal: "`s0.0.0-0.0.2`" (dos elementos) o "`s0.0.0`" (operación entera).
- Factor común: commonfactor. Saca factor común, ya sea un número entero o una fracción. Posibilidad de especificar el factor. Terminal: "`f0.0.0`" o "`f0.0.0 1/3`".
- Eliminar factor común en ambos miembros: delcommonfactor. Terminal: "del".

D

Ecuación en formato árbol

En esta sección se muestran los pasos que realiza ChalkPy para resolver una ecuación de forma automática a través del ejemplo utilizado en la memoria.

La definición de las acciones se hace a través de la siguiente sintaxis:

- s[id1]-[id2]. Opera los términos con los identificadores id1 e id2.
- m[id1]-[id2]. Mueve el término con identificador id1 a la posición del término id2.
- f[id1]-[id2] x. Factoriza los términos con identificadores id1 e id2. Si se especifica x, se utiliza dicho factor.
- d[id1]-[id2]. Elimina los términos con identificador id1 e id2 si coinciden.

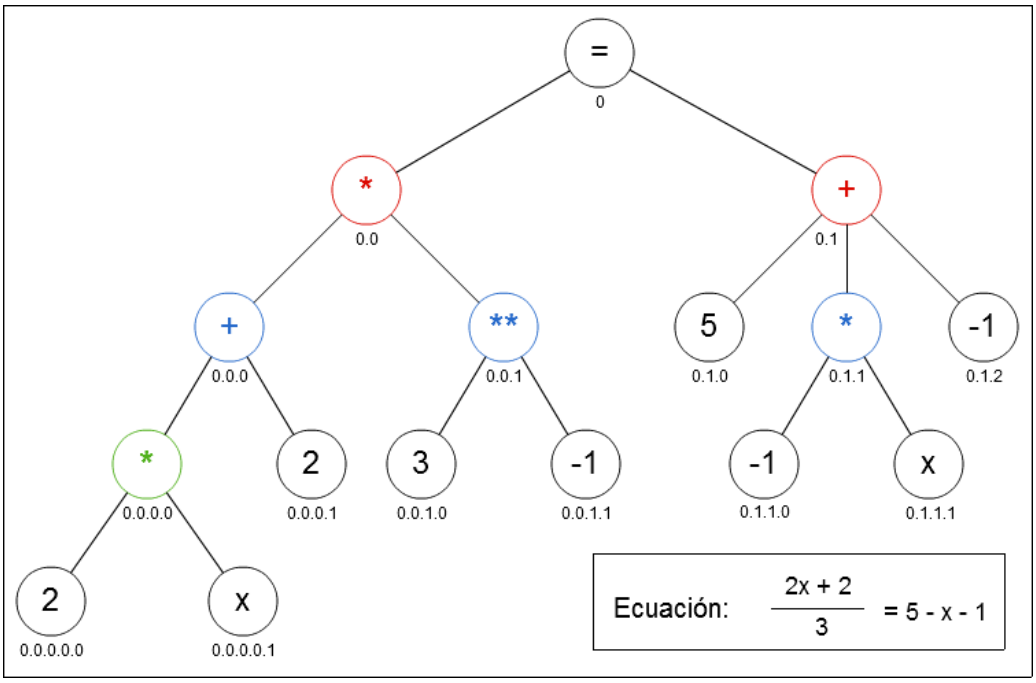

Figura D.1: Expresión `(2x+2)/3=5-x-1` expresada como árbol

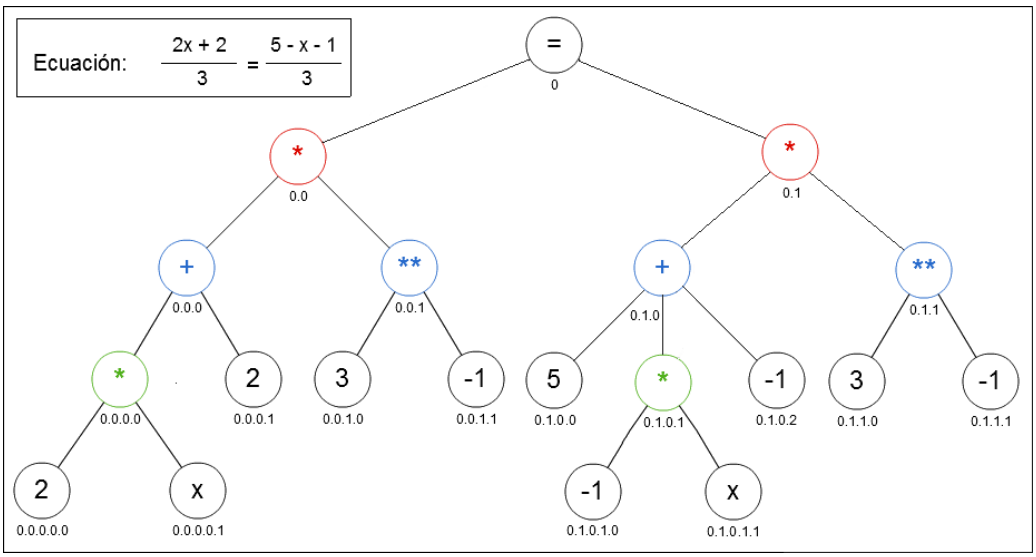

Figura D.2: Expresión tras la operación `f0.1.0-0.1.1-0.1.2 1/3`

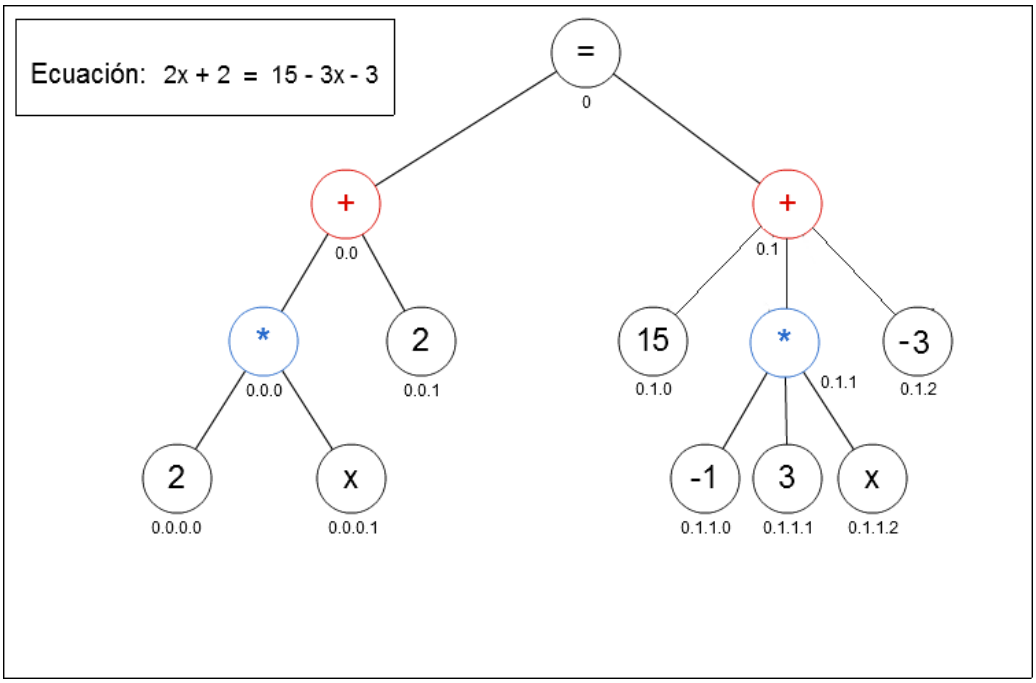

Figura D.3: Expresión tras la operación `d0.0.1-0.1.1`

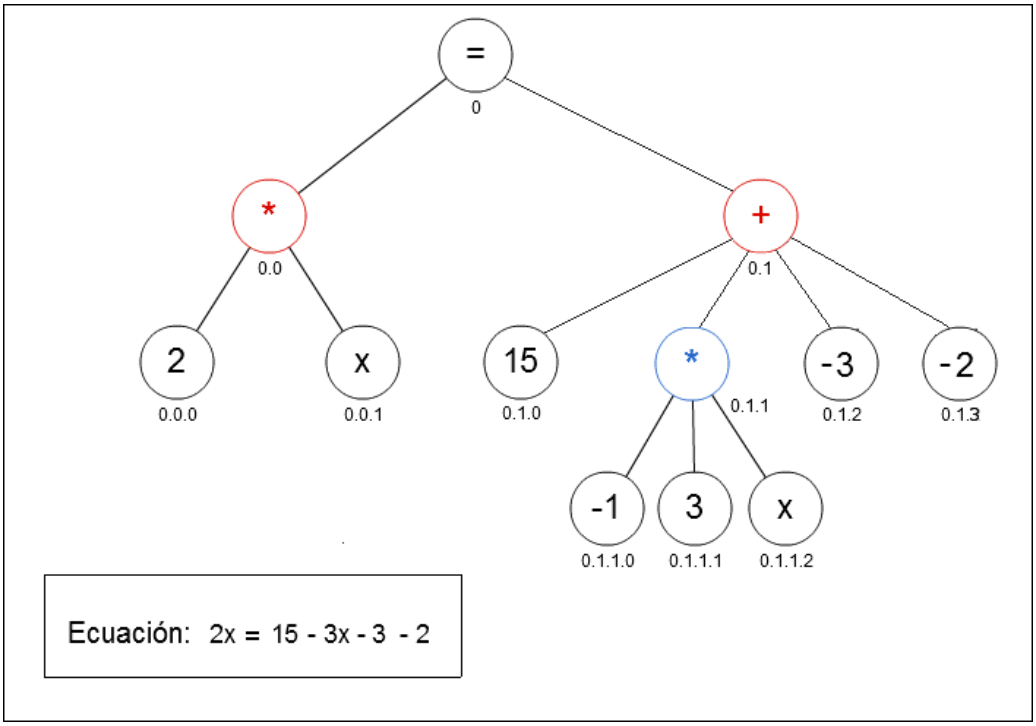

Figura D.4: Expresión tras la operación `m0.0.1-0.1.2`

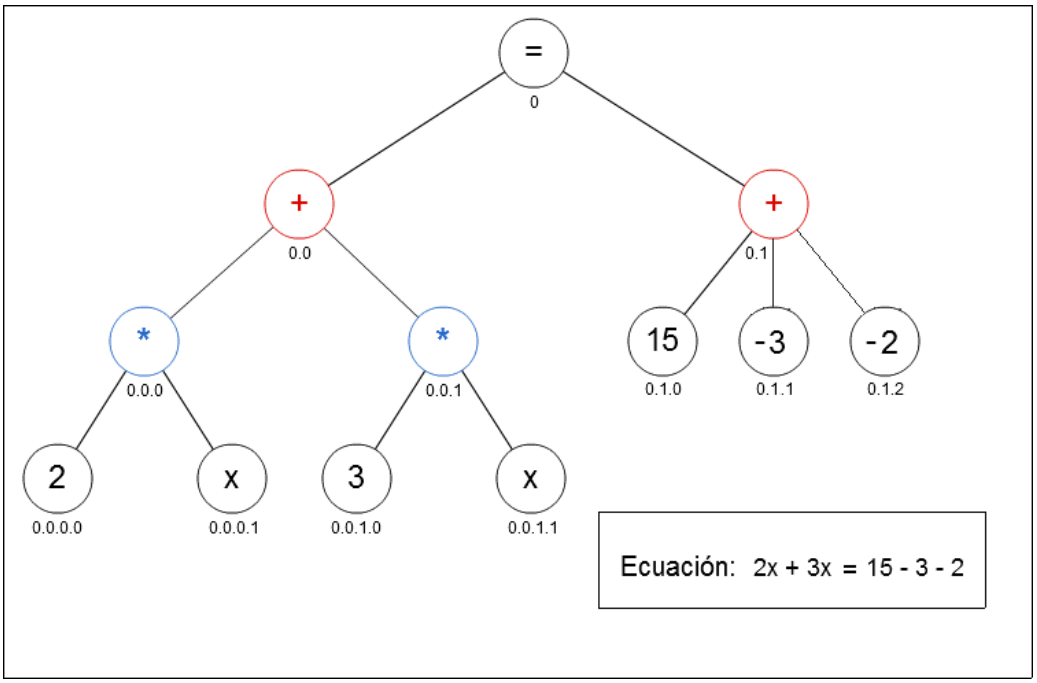

Figura D.5: Expresión tras la operación `m0.1.1-0.0`

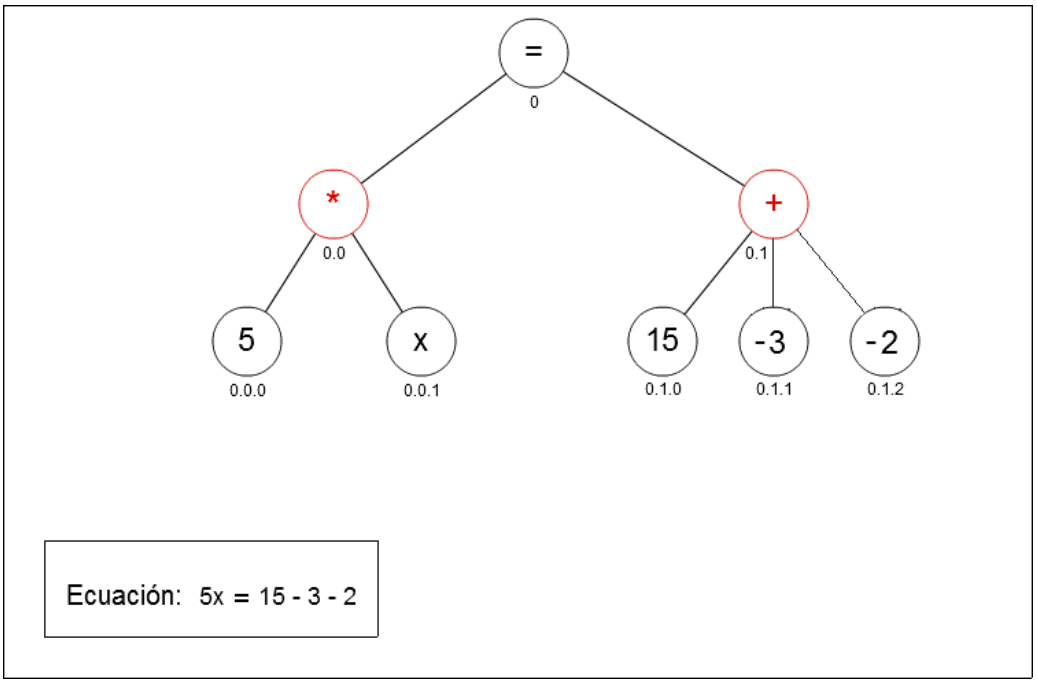

Figura D.6: Expresión tras la operación `s0.0.0-0.0.1`

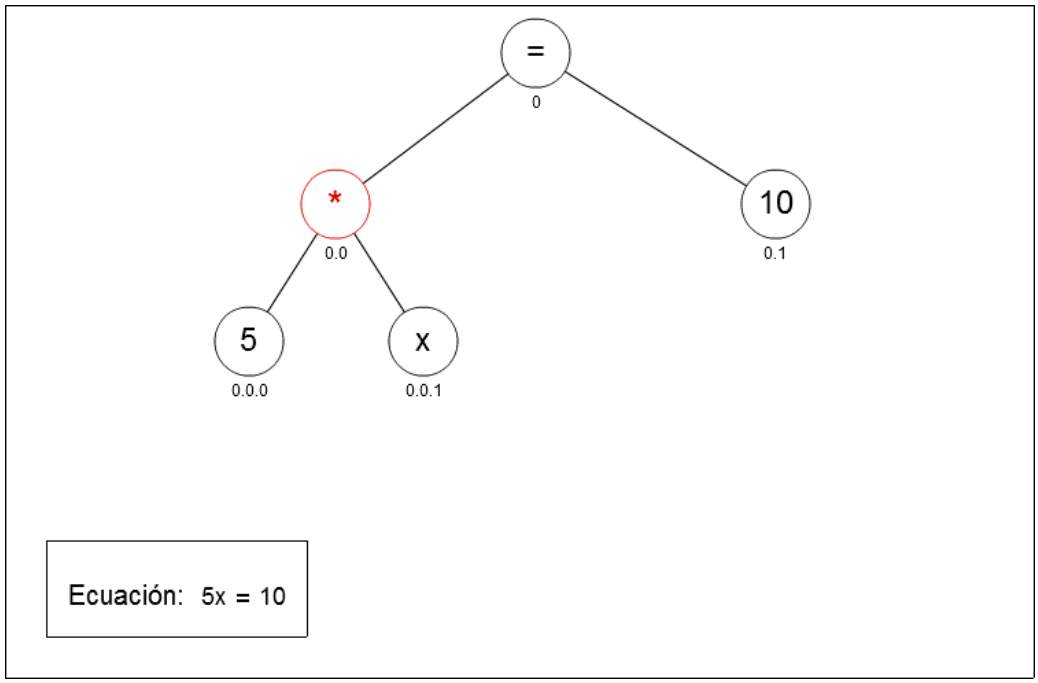

Figura D.7: Expresión tras la operación `s0.1.0-0.1.1-0.1.2`

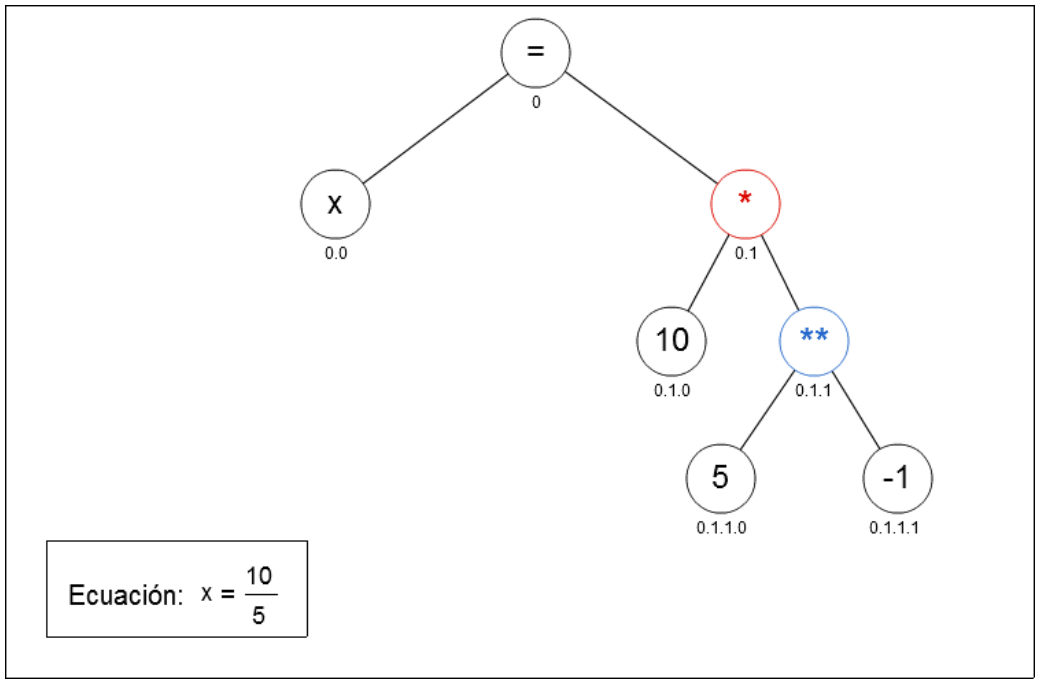

Figura D.8: Expresión tras la operación `m0.0.0-0.1`

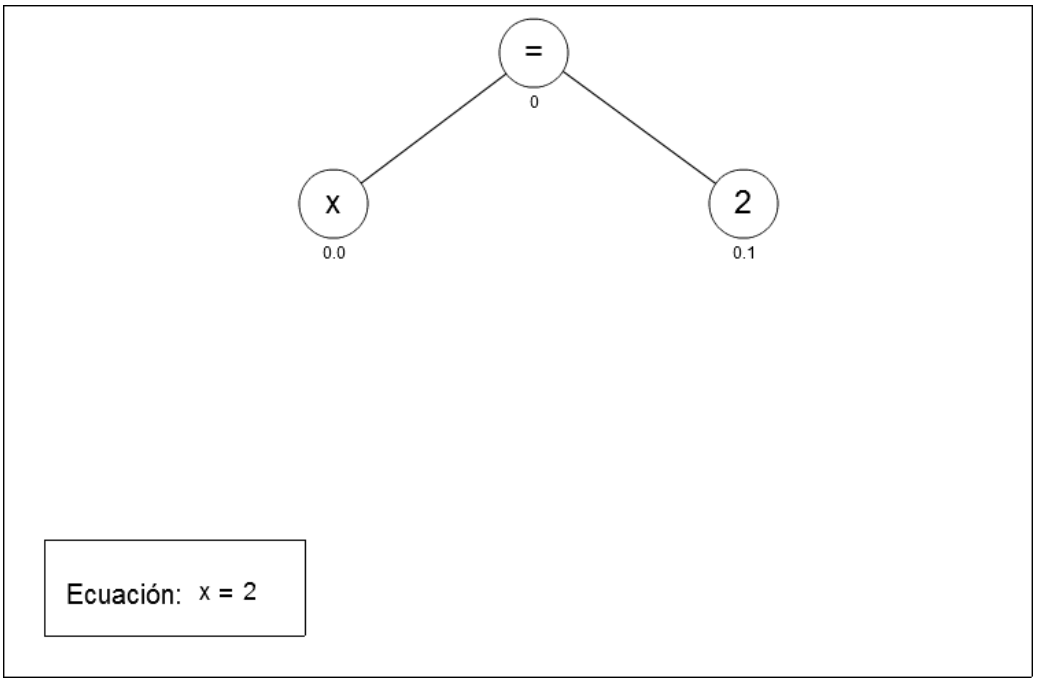

Figura D.9: Expresión tras la operación `s0.1.0-0.1.1`

E

Pruebas a usuarios

A continuación se muestra la información obtenida sobre las distintas pruebas realizadas a los diferentes usuarios de ChalkPy.

Prueba alfa

La prueba alfa se realizó sobre diez usuarios con distintos perfiles altos y bajos en relación a las matemáticas y al manejo de aplicaciones web.

Todos los usuarios coincidieron en que la aplicación es útil para practicar la resolución de ecuaciones y no encontraron dificultad en su manejo.

Algunos comentarios que engloban la opinión de los usuarios y sus recomendaciones:

- *Creo que resulta fácil y dinámico para un profesor de la ESO utilizar la herramienta para las sesiones presenciales, incluso para solicitar la participación de los alumnos y exponer la distintas formas de resolución de las ecuaciones.*
- *La herramienta es sencilla pero funcional para enseñar y realizar ecuaciones a nivel de 1º de la ESO.*
- *ChalkPy es bastante cómoda para resolver ecuaciones y poder revisar los pasos realizados.*
- *Es muy interesante que muestre el registro de cada operación al lado de la pizarra y la opción de guardar el documento.*

- *La resolución automática es muy útil para contrastar las distintas formas de resolver la ecuación.*
- *La interfaz es muy agradable, los colores adecuados y los iconos están bien identificados.*
- *El icono de resolución de la ecuación no es descriptivo.*
- *Me gustaría operar arrastrando un número sobre otro.*
- *Se echa en falta una notificación al resolver la ecuación.*

Prueba beta

La prueba alfa se realizó a los profesores y alumnos del Máster Universitario en Formación de Profesorado de Educación Secundaria Obligatoria y Bachillerato de la rama de Matemáticas y a varios estudiantes de la ESO. El objetivo principal de la prueba a los docentes era conocer su interés por el uso de nuevas tecnologías en el aula y si consideran que ChalkPy cubre las desventajas del uso de una pizarra clásica.

Estas fueron las preguntas realizadas en una encuesta a los trece docentes que participaron:

1. ¿Considera importante la introducción de nuevas tecnologías en el aula? Resultados en gráfica E.1.
2. ¿Le parece ChalkPy un programa adecuado para introducir los conceptos básicos de las ecuaciones algebraicas? Resultados en gráfica E.2.
3. ¿Como profesor, utilizaría ChalkPy en su clase? Resultados en gráfica E.3.
4. ¿Considera ChalkPy útil para el estudio individual de los estudiantes? Resultados en gráfica E.4.
5. ¿Le resulta fácil de usar ChalkPy? Resultados en gráfica E.5.
6. Señale con qué tecnología preferiría utilizar ChalkPy en clase. Resultados en gráfica E.6.

La conclusión de estas preguntas junto con los comentarios realizados durante la prueba es que los docentes sí están interesados en una herramienta como ChalkPy para cubrir el área del álgebra, puesto que les facilitaría el proceso de enseñanza.

Un aspecto interesante de esta prueba es el alto interés que mostraron por el uso de la aplicación sobre la pizarra digital. Esto se debe a que durante la prueba se comprobó que estar sentado delante del ordenador explicando los pasos no era tan dinámico ni captaba tanto la atención como interactuar con la ecuación con los propios dedos en la pizarra digital.

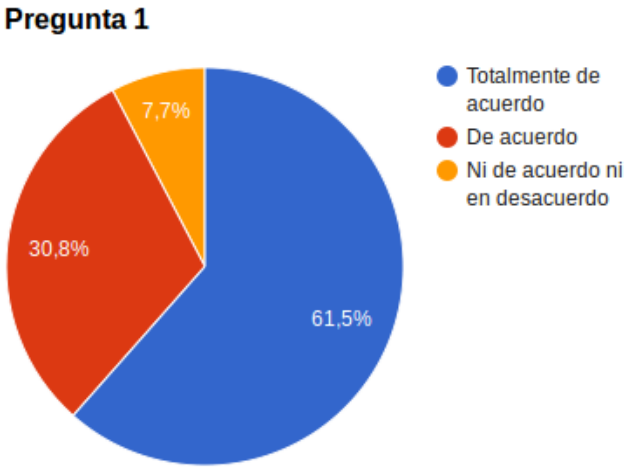

Figura E.1: ¿Considera importante la introducción de nuevas tecnologías en el aula?

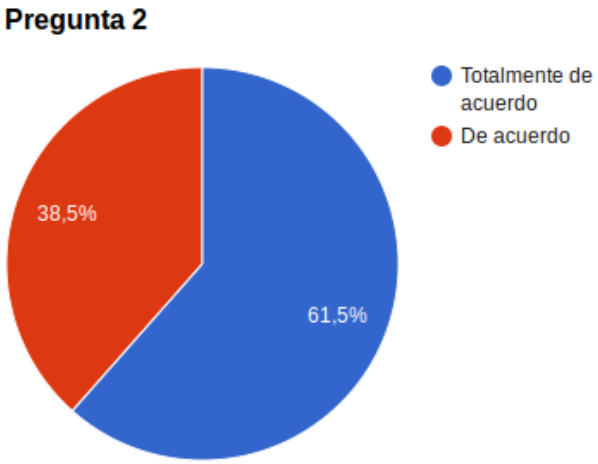

Figura E.2: ¿Le parece ChalkPy un programa adecuado para introducir los conceptos básicos de las ecuaciones algebraicas?

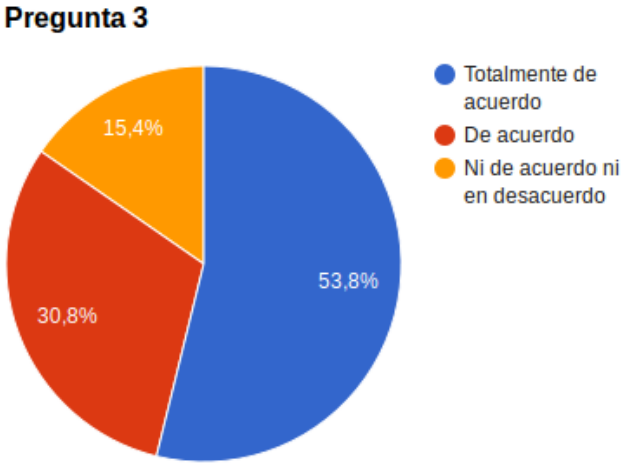

Figura E.3: ¿Como profesor, utilizaría ChalkPy en su clase?

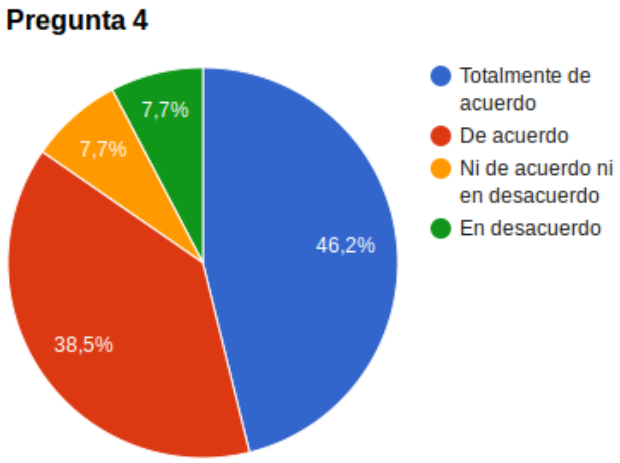

Figura E.4: ¿Considera ChalkPy útil para el estudio individual de los estudiantes?

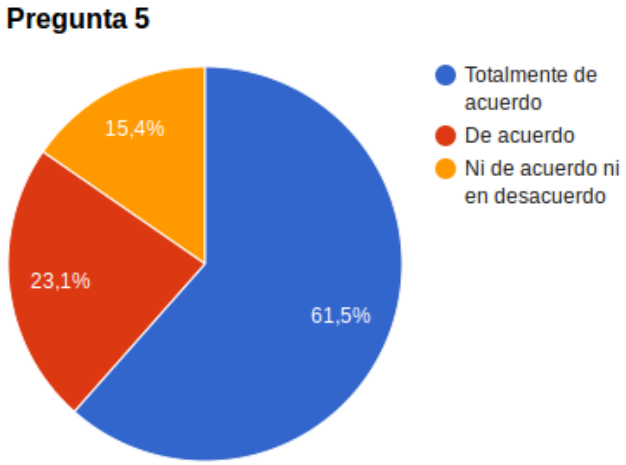

Figura E.5: ¿Le resulta fácil de usar ChalkPy?

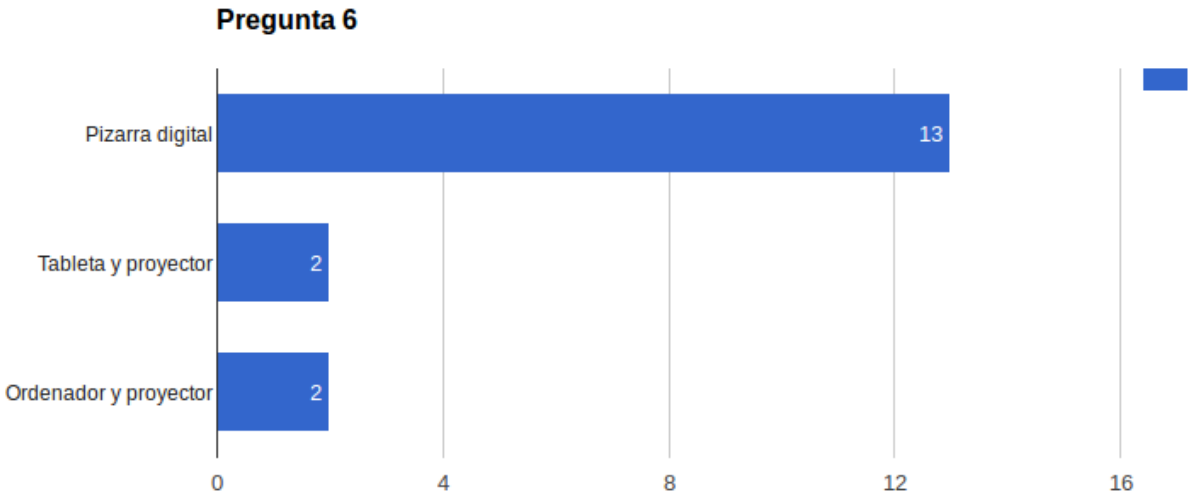

Figura E.6: Señale con qué tecnología preferiría utilizar ChalkPy en clase.

Algunos comentarios que engloban la opinión de los usuarios y sus recomendaciones:

- *Me ha parecido una herramienta muy útil para utilizarla en clase. Creo que sería muy interesante implementarla para más aspectos del temario de Matemáticas.*
- *El botón de resolver lo dejaría sólo para usuario-profesor, no para alumnos.*
- *Una versión móvil para que los alumnos practiquen haciendo ejercicios. Añadir algún tipo de juego que dé puntos cuando hacen bien las ecuaciones.*
- *Cambiaría opciones de doble clic por arrastrar un número sobre otro.*
- *La pizarra digital la veo útil.*

Por su parte, los tres estudiantes de la ESO a los que se cuestionó mostraron interés por la herramienta en clase, principalmente para su uso en caso de tener que corregir un ejercicio delante de sus compañeros.

Algunos de sus comentarios fueron:

- *Creo que el programa está bien porque nos ahorra tiempo de explicación y con el tiempo restante te puedes dedicar a hacer más ejercicios.*
- *Lo que más me gusta es que resuelva la ecuación solo. Me gustaría tener algo así en mi cuaderno.*
- *Estaría bien para corregir ejercicios pero para practicar es mejor hacerlos a mano, porque así haces tú las cuentas.*

F

Control de recursos

A continuación se indican las fuentes de los recursos utilizados en ChalkPy. Se ha tenido especial cuidado en que fuesen de dominio público o estuviesen bajo la licencia Creative Commons.

- Imágenes textura pizarra, por *Nick Georgiou* [50].
- Imagen cuaderno. Modificación de *spiral bound writing pad*, por *Creativity103* [51].
- Diseño HTMLs con Ion [52] y Entryway [53], por *templated.co* [54].
- Imagen alumno resolviendo una ecuación en la pizarra. *Students study an equation on a chalk board*, por *simpleinsomnia* [55].
- Imagen check verde, de Wikimedia [56].
- Imagen delete, de Wikimedia [57].
- Imagen add (+), de Wikimedia [58].
- Imagen undo, de Wikimedia [59].
- Imagen redo, de Wikimedia [60].
- Imagen reload, de Wikimedia [61].
- Imagen save, de Wikimedia [62].
- Imagen solve, de Wikimedia [63].
- Imagen PDF, de Wikimedia [64].